The Geometry of Generative Models

Kalatzis, Dimitrios

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

DTU Compute
Institut for Matematik og Computer Science

The Geometry of Generative Models

Dimitris Kalatzis

Supervisor: Prof. Søren Hauberg
Co-supervisor: Prof. Ole Winther

DTU Compute

Section for Cognitive Systems

Technical University of Denmark

Richard Petersens Plads

Building 321

2800 Kongens Lyngby, Denmark

Summary

Deep generative models have achieved remarkable success in modelling various types of data, such as images or natural language. In most cases these types of data are considered to be Euclidean or are being modelled with Euclidean tools. Many types of data, however, are most "naturally" represented on non-Euclidean manifolds or are considered to reside on some lower dimensional non-Euclidean manifold embedded in a Euclidean ambient space. Such cases lead to known failure modes of many popular deep generative models. In response, there has been a recent interest and effort in developing generative models with structural priors tailored for specific manifolds. This thesis is dedicated to using tools from differential geometry and topology to develop generative models for efficiently modelling manifold-valued data with no assumptions on the topological properties on the underlying manifold structure of the data.

In chapter 2, we focus on the estimation of geodesic distances and pull-back metrics in the context of variational autoencoders to preserve relationships between data points and subsequently make the associated latent spaces identifiable and informative with regards to the geometric structure of the data. In chapter 3 we generalize this scheme to variational autoencoders with a variety of non-Gaussian decoders. Finally, in chapter 4 we show that we can take advantage of the class of functions represented by normalizing flows to build generative models that form a smooth atlas over the data manifold, thus using locally Euclidean tools to learn the overall non-Euclidean structure of the data. We evaluate these methods over a range of tasks from density estimation of synthetic, image, geological and physical systems data to downstream tasks such as classification, pose estimation and posterior inference.

Preface

This thesis was prepared at the section for Cognitive Systems in the Department of Applied Mathematics and Computer Science (DTU Compute), Technical University of Denmark. It constitutes a partial fulfillment of the requirements for acquiring a PhD at the Technical University of Denmark.

The PhD project was supervised by Søren Hauberg and Ole Winther and it was financed by the EU2020 project "Measuring with no tape". The PhD project was carried out at the Technical University of Denmark from November 2018 to August 2022 with the following exceptions: an external research stay at the University of Amsterdam (AMLAB), Amsterdam, the Netherlands from February 2020 to June 2020, where I was supervised by Patrick Forré, and an internship at Qualcomm AI Research in Amsterdam, the Netherlands from May 2021 to November 2021, where I was supervised by Arash Behboodi.

Throughout my PhD, I have worked on applying principles from topology and differential geometry to generative models, specifically variational autoencoders and normalizing flows. This framework was applied to both synthetic and real world use cases ranging from human pose data to geological and physical systems data. This has resulted in three research papers, two of which are peer-reviewed and one currently being considered for publication, all of which, constitute the material for this thesis.

Kongens Lyngby, 1st September 2022

Dimitris Kalatzis

Acknowledgements

First and foremost, I would like to thank my supervisors Søren Hauberg and Ole Winther. Søren has taught me the value of mathematical rigour and good intuition, but also optimism in research, though I've yet to fully internalize that last part. More importantly though, Søren gave me a chance to learn how to conduct research back in 2018 and has repeatedly gone above and beyond the call of duty of a supervisor throughout all these years, for which I will be eternally grateful to him. I had fewer interactions with Ole throughout my PhD, but they were all very fruitful and helped shape my habits and identity as a researcher for which I'm thankful.

I'd also like to thank all my friends, mentors and collaborators at AMLAB and Qualcomm AI Research from which I have learned a lot: Patrick Forré, Arash Behboodi, Johann Brehmer, Daniel Dijkman, Hanno Ackermann, Tribhuvanesh Orekondy, Ilia Karmanov, Farhad Ghazvinian Zanjani and Christos Louizos.

Furthermore, I'd like to thank my collaborators Jesper Wohlert, David Eklund, Johan Ye, Miguel González-Duque, Alison Pouplin, Georgios Arvanitidis and Søren Hauberg for working with me and contributing to my training as a researcher, each in their own special way.

I am grateful to my friends and officemates in Copenhagen and elsewhere, Didrik Nielsen, Maxim Khomiakov, Giorgio Giannone, Valentin Liévin, Cilie Feldager Hansen for many fun times and fruitful discussions, Siobhan McKenzie Hall for being a true friend and whose perseverance and determination has been an inspiration ever since I met her. Also, Georgios Arvanitidis for his help and support upon my arrival in Copenhagen back in 2018, and Pablo Moreno Muñoz, Pola Schwöbel and Federico Bergamin who will never know how much their company and support meant to me during the final year of my PhD.

I also want to express my gratidute to Nikos who has been my friend and comrade for more than 20 years and our friendship is one of these things that has kept me sane throughout my PhD and Vassilis, whose company, cigarettes and discounted drinks have also provided help in the same way. Also Eve for her love, support and patience with me throughout my PhD. I'm also grateful to my family for all their love and support and for cultivating my curiosity and faith in hard work.

Finally, I'm grateful for every obstacle in my path presented by people or simply

unlucky circumstances. One can expand the limits of one's strength and resilience only through adversity.

Contents

CHAPTER 1

Introduction

The *manifold hypothesis* is a common heuristic in machine learning which states that data lie near or on a lower dimensional manifold embedded in some high dimensional ambient space. It is a useful heuristic, underlying the bulk of dimensionality reduction techniques, which are so prevalent in machine learning. It is also a central premise to the development of the models presented in subsequent chapters of this thesis. Thus, to make subsequent exposition clearer we will attempt to demystify the notion of a manifold from a topological and geometric perspective in the section that immediately follows. Then we will conclude this introductory chapter by reviewing the details of the generative models present in this thesis.

1.1 The topology of smooth manifolds

Let's begin with the definition of a smooth manifold.

Definition 1. *A smooth manifold $\mathcal{M}$ of dimension d is a topological space that is locally Euclidean, i.e. each point of $\mathcal{M}$ has a neighborhood U which is diffeomorphic to an open subset of $V \subset \mathbb{R}^d$.*

We can now formalize the "locally Euclidean" property of a smooth manifold by introducing *smooth local coordinate charts* on $\mathcal{M}$.

Definition 2. *Given a d-dimensional topological manifold $\mathcal{M}$, a smooth coordinate chart on $\mathcal{M}$ is a pair (U, ϕ), where $\phi\colon U \to V$ is a diffeomorphism between the open subsets $U \subset \mathcal{M}$ and $V \subset \mathbb{R}^d$.*

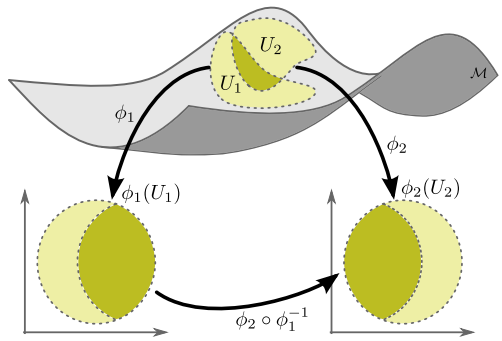


Figure 1.1. Smoothly compatible charts.

To have local coordinates for every point on $\mathcal{M}$ we can define a collection of smooth coordinate charts that covers $\mathcal{M}$. This collection is called a *smooth atlas*. This construction is necessary to define smooth functions (such as probability density

functions) and perform gradient-based optimization on $\mathcal{M}$, since for any smooth coordinate chart (U, ϕ) and a function $f : \mathcal{M} \to \mathbb{R}$, the composition $f \circ \phi^{-1} : V \to \mathbb{R}$ is smooth. It further allows us to account for points occurring in overlapping charts without issues with regard to smoothness, since given two smooth coordinate charts $(U_1, \phi_1), (U_2, \phi_2)$ with $U_1 \cap U_2 \neq \emptyset$, the composition $\phi_2 \circ \phi_1^{-1}$ is smooth and invertible. These charts are then called *smoothly compatible* (see Fig. 1.1).

In this work we are considering a smooth manifold $\mathcal{M}$ of dimension d, embedded in some Euclidean space $\mathbb{R}^D$ with $d < D$. Embedded submanifolds can be defined as the images of *smooth embeddings*.

Definition 3. *A smooth embedding is a smooth immersion (i.e. a map, with Jacobian that is full rank everywhere), which is also a diffeomorphism onto its image.*

More specifically, a neighborhood $U \subset \mathcal{M}$ can be expressed as the image of a smooth embedding $F : V \to \mathbb{R}^D$, with $V \subset \mathbb{R}^d$ (see Fig. 1.2). Smooth embeddings are diffeomorphisms onto their image and as such, invertible when their codomain is restricted to it. Thus, the open subset $U \subset \mathcal{M}$ inherits the Euclidean topology of V and we can define local coordinates on U, through the coordinate chart (U, ϕ) with $\phi = F^{-1} : \mathbb{R}^D \to V$ by restricting the domain of F^{-1} to U.

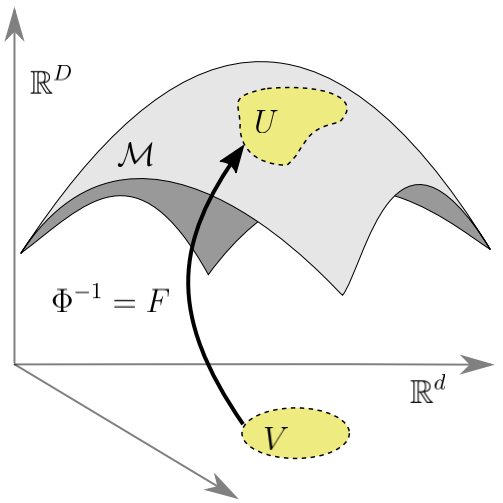


Figure 1.2. A neighborhood U of an embedded submanifold $\mathcal{M} \subset \mathbb{R}^D$ is the image of a smooth embedding $F : V \to \mathbb{R}^D$, with $V \subset \mathbb{R}^d$.

1.2 A review on Riemannian geometry

Hence, we give a short review of Riemannian geometry.

A *smooth manifold* $\mathcal{M}$ is a topological manifold endowed with a smooth structure. That is to say $\mathcal{M}$ is locally homeomorphic to Euclidean space and we are able to do calculus on it. For a point $p \in \mathcal{M}$, the *tangent space* $T_p\mathcal{M}$ is a vector space centered on p which contains all tangent vectors to $\mathcal{M}$ passing through point p (Fig. 1.3). With this we can give a formal definition of the Riemannian metric tensor which is of central importance to any analysis involving Riemannian geometry.

Definition 4. ***(Riemannian metric)*** *[do Carmo, 1992] Given a smooth manifold $\mathcal{M}$, a Riemannian metric on $\mathcal{M}$ assigns on each point $p \in \mathcal{M}$ an inner product (i.e. a symmetric, positive definite, bilinear form) $\langle \cdot, \cdot \rangle_p$ in the tangent space $T_p\mathcal{M}$ which varies smoothly in the following sense: if $\mathbf{x} : \mathbb{R}^n \supset U \to \mathcal{M}$ is a local coordinate chart centered at p and $\frac{\partial}{\partial x_i}(q) = \mathbf{dx}_q(0, \ldots, 1, \ldots, 0)$ for $q \in U$, then $\langle \frac{\partial}{\partial x_i}(q), \frac{\partial}{\partial x_j}(q) \rangle_{\mathbf{x}(q)} = g_{ij}(q)$ is a smooth function on U.*

By generalizing the inner product to Riemannian manifolds, the metric tensor gives meaning to length, angle and volume on manifolds. Central to distributions defined on a Riemannian manifold, the volume measure over an infinitesimal region centered at point p is defined as $d\mathcal{M}_p = \sqrt{\det \mathbf{G_p}} dp$, where G_p is the matrix representation of the metric tensor evaluated at point p. Shortest paths on manifolds are represented by geodesic curves, which generalize straight lines in Euclidean space. A geodesic is a constant speed curve and its length can be computed by integrating the norm of its velocity vector under the metric, in other words $\mathcal{L} = \int_0^1 ||\frac{d\gamma}{dt}||_g dt$. For $p \in \mathcal{M}$ there is a useful map defined on a neighborhood of the origin of $T_p\mathcal{M}$ called the exponential map. More precisely, the exponential map is a diffeomorphism, i.e. a smooth map with a smooth inverse, between an open subset $\mathcal{U} \subset T_p\mathcal{M}$ and an open subset $\mathcal{U}' \subset \mathcal{M}$. Given $p \in \mathcal{M}$ and $v \in \mathcal{U}$, there is a unique geodesic $\gamma : [0, 1] \to \mathcal{M}$ with $\gamma(0) = p$ and $\frac{d\gamma}{dt}(0) = v$. The exponential map is given by $exp_p(v) = \gamma(1)$. Note that $exp_p(0) = p$. The inverse map (from $\mathcal{U}'$ to $\mathcal{U}$) $exp_p^{-1} = log_p$ is called the *logarithmic map*.

Let $\mathcal{M} \subseteq \mathbb{R}^M$ be an embedded n-dimensional manifold and consider local coordinates $\phi : U \to \mathcal{M}$ with $U \subseteq \mathbb{R}^N$ an open subset. The Euclidean metric on $\mathbb{R}^M$ induces a Riemannian metric on $\mathcal{M}$. Expressed in terms of the coordinates given by ϕ, this metric is known as the *pull-back metric* on U under ϕ. For $\mathbf{u} \in U$, the pull-back metric $\mathbf{G_u}$ at $\mathbf{u}$ is given by

$$\mathbf{G_u} = J_\phi^\top(\mathbf{u}) J_\phi(\mathbf{u}), \tag{1.1}$$

where J_ϕ denotes the Jacobian matrix of ϕ.

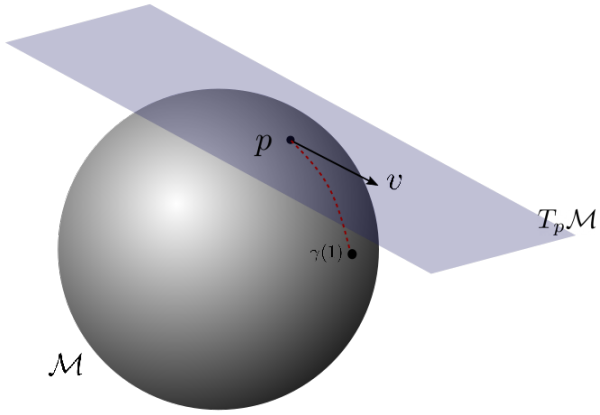


Figure 1.3. A manifold $\mathcal{M}$ with a tangent space $T_p\mathcal{M}$ centered at point p. The exponential map centered at p, maps the tangent vector $v \in T_p\mathcal{M}$ to the (red, dashed) geodesic curve on the manifold $\mathcal{M}$.

1.3 Normalizing flows

1.3.1 Definition for "discrete time" flows

A normalizing flow [Rezende and Mohamed, 2015] consists of a diffeomorphic map $f_\theta : \mathcal{U} \to \mathcal{X}$ parameterized by θ with $\mathcal{U} \subseteq \mathbb{R}^D$ and $\mathcal{X} \subseteq \mathbb{R}^D$, and a base or "latent" distribution on $\mathcal{U}$, with density given by $p(\boldsymbol{u})$. Typically, the goal is to estimate a target probability density on $\mathcal{X}$ given by $p^\star(\boldsymbol{x})$, which is usually achieved through maximum likelihood estimation of a model likelihood $p(\boldsymbol{x})$. We assume the following generative process:

$$\begin{aligned} \boldsymbol{u} &\sim p(\boldsymbol{u}) \\ \boldsymbol{x} &= f(\boldsymbol{u}; \theta), \end{aligned} \tag{1.2}$$

where the parameters of the flow, θ are given by neural networks.

The map f induces a probability density in $\mathcal{X}$ and since it is a diffeomorphism (a differentiable, invertible map with a differentiable inverse) we can evaluate $p(\boldsymbol{x})$ *exactly* using the change of variables formula:

$$p(\boldsymbol{x}) = p(\boldsymbol{u}) |\det J_f(\boldsymbol{u})|^{-1} \tag{1.3}$$

$$= p(\boldsymbol{u}) |\det J_{f^{-1}}(\boldsymbol{x})| \tag{1.4}$$

with $\boldsymbol{u} = f^{-1}(\boldsymbol{x})$.

In practice, the diffeomorphism f is constructed as a composition of simpler diffeomorphisms and in a subsequent section we will see some ways this is implemented in practice. Intuitively, this construction describes the discretized dynamics of f in as many steps as we have compositions. This intuition gives rise to an alternative way of considering f, namely in the "continuous time" setting, where f is constructed through the parameterization of its infinitesimal dynamics. As such, f is now defining a vector field over the data space $\mathcal{X}$, which we need to integrate to retrieve the transformation from the target distribution to the base distribution. Thus, a continuous normalizing flow (CNF) is an integral curve given by an ordinary differential equation (ODE). In this thesis we will focus on discrete flows, however we include the definition for CNFs for completeness in the next section.

1.3.2 Definition for "continuous time" flows

We denote by $\boldsymbol{z}_t$ the dynamics of the flow at time t with $t \in [t_0, t_1]$. We assume that $\boldsymbol{z}_{t_0} = \boldsymbol{u}$ and $\boldsymbol{z}_{t_1} = \boldsymbol{x}$, with $\boldsymbol{u}, \boldsymbol{x} \in \mathbb{R}^D$. A CNF [Chen et al., 2018c] parameterizes the derivative $\frac{d\boldsymbol{z}_t}{dt}$ with a function f_θ, itself parameterized by θ:

$$\frac{d\boldsymbol{z}_t}{dt} = f_\theta(\boldsymbol{z}_t, t) \tag{1.5}$$

The function f_θ needs to be Lipschitz continuous for the above ODE to have a unique solution according to the Picard-Lindelöf theorem [Coddington and Levinson, 1955]. In practice, f_θ is implemented as a neural network which is Lipschitz continuous. To compute the forward transformation, we integrate the dynamics from t_0 to t_1:

$$\boldsymbol{z}_{t_1} = \boldsymbol{u} + \int_{t_0}^{t_1} f_\theta(\boldsymbol{z}_t, t)dt \tag{1.6}$$

To compute the inverse transform, we integrate in the reverse direction:

$$\boldsymbol{z}_{t_0} = \boldsymbol{x} + \int_{t_1}^{t_0} f_\theta(\boldsymbol{z}_t, t)dt \tag{1.7}$$

In the "discrete time" case the change in (log) density is equivalent to the change in volume $|\det J_f(\boldsymbol{u})|^{-1}$. The "continuous time" equivalent is given by Chen et al. [2018c]:

$$\frac{d\log(\boldsymbol{z}_t)}{dt} = -\operatorname{Tr}(J_f(\boldsymbol{z}_t)), \tag{1.8}$$

where $\operatorname{Tr}(\cdot)$ denotes the trace of a square matrix and $J_f(\boldsymbol{z}_t)$, the Jacobian of f. In practice, to avoid having to make $\mathcal{O}(D)$ backpropagation calls, the trace can be

estimated by Hutchinson's estimator [Hutchinson, 1989], which was first introduced in a Neural ODE/CNF setting by Grathwohl et al. [2018]:

$$\mathrm{Tr}(J_f) = \mathbb{E}_{p(\epsilon)}[\epsilon^\top J_f \epsilon], \tag{1.9}$$

where ϵ is a random vector with $\mathbb{E}[\epsilon] = 0$ and $\mathrm{Cov}[\epsilon] = I$. Finally, the log density of $\boldsymbol{x}$ can be computed as:

$$\log p(\boldsymbol{x}) = \log p(\boldsymbol{u}) - \int_{t_0}^{t_1} \mathrm{Tr}(J_f(\boldsymbol{z}_t))dt \tag{1.10}$$

1.3.2.1 Backpropagation with the Adjoint Method

The adjoint sensitivity method [Pontryagin, 1987] was used by Chen et al. [2018c] as an alternative to backpropagation through the ODE solver, which is computationally inefficient. They show that the gradient of the loss with respect to the intermediate states $\boldsymbol{z}_t$ is given by the following ODE:

$$\frac{d}{dt}\frac{\partial L}{\partial \boldsymbol{z}_t} = -\frac{\partial L}{\partial \boldsymbol{z}_t}^\top \frac{\partial f_\theta(\boldsymbol{z}_t, t)}{\partial \boldsymbol{z}_t} \tag{1.11}$$

The quantity $\frac{\partial L}{\partial \boldsymbol{z}_t}$ is called the adjoint state of the ODE. The gradient with respect to flow parameters θ is given by:

$$\frac{dL}{d\theta} = \int_{t_1}^{t_0} \frac{\partial L}{\partial \boldsymbol{z}_t}^\top \frac{\partial f_\theta(\boldsymbol{z}_t, t)}{\partial \theta} dt \tag{1.12}$$

The forward pass 1.6 is computed by a call to an ODE solver, while another call to the solver will yield the gradient with respect to the flow parameters 1.12. This process can be considered the continuous time analog to the backpropagation algorithm [Rumelhart et al., 1986a].

1.3.3 Construction of flow models

Throughout this section we will discuss some of the most common transformations used in the construction of flow models. We will focus on transformations that are used throughout this work. For a more exhaustive overview we refer the interested reader to Papamakarios et al. [2021].

An obvious consideration in the design of normalizing flows is dealing effectively with the determinant of the Jacobian of f. Taking the determinant of arbitrary matrices is

an $\mathcal{O}(D^3)$ operation which in all but the most trivial cases is prohibitively expensive and so a naïve call to a determinant method is impractical. Furthermore, we would like to be able to invert f efficiently - ideally by having access to the analytical inverse, which would enable efficient maximum likelihood training. Thus, the flow transformations presented below are designed with two goals in mind:

1. Reducing the computational cost of computing the determinant while paying the minimum price in regard to the overall transformation flexibility.
2. Inverting the overall transformation (ideally) in constant time.

The main strategy to achieve both of these objectives is to design the transformation f as a composition of simple transformations g_i, which are easily invertible and their Jacobian determinants are cheap to compute. Thus, the transformation f has the following form:

$$f(\boldsymbol{x}) = g_N(\boldsymbol{z}_{N-1}) \circ g_{N-1}(\boldsymbol{z}_{N-2}) \circ \cdots \circ g_1(\boldsymbol{x}) \tag{1.13}$$

where N is the number of transformations used in the composition and $\boldsymbol{z}_i$ are the intermediate representations.

The Jacobian of f is of the form:

$$J_f = J_{g_N} \cdot J_{g_{N-1}} \cdot \ldots \cdot J_{g_1} \tag{1.14}$$

And for the computation of the determinant of J_f we have:

$$\det J_f = \det\big(J_{g_N} \cdot J_{g_{N-1}} \cdot \ldots \cdot J_{g_1}\big) \tag{1.15}$$

$$= \det J_{g_N} \det J_{g_{N-1}} \ldots \det J_{g_1} \tag{1.16}$$

We note that depending on the application, the two objectives/restrictions to the design of flow transformations mentioned above regarding tractability of determinants and access to the analytical inverse of the transformation can be relaxed. E.g. Ho et al. [2019], Wehenkel and Louppe [2019] use transformations which are guaranteed to be invertible but do not have an analytical inverse. In such cases, root finding methods can be employed with the most popular one being bisection [Burden et al., 2015]. For a more comprehensive overview of such methods we once again refer the interested reader to Papamakarios et al. [2021].

1.3.3.1 Invertible linear transformations

Given $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^D$, linear transformations have the following form:

$$\boldsymbol{z} = g(\boldsymbol{x}) \tag{1.17}$$

$$= W\boldsymbol{x}, \tag{1.18}$$

with $W \in \mathbb{R}^{D \times D}$

The simplest way to implement this transformation would be to directly parameterize the matrix W. However, for our transformation to be practically useful, we need to consider the two objectives/restrictions we laid out in the previous section. Let's first consider the matter of computing the determinant of the Jacobian of g. The Jacobian of g in this case is the matrix in question itself, W. We have already stated in the previous section that taking the determinant of an arbitrary matrix is too costly to be practical. As for the second restriction, that pertaining to the invertibility of g, the matrix W is not guaranteed to be invertible, at least in this general way we have defined it. Even if it is invertible, inverting this matrix naïvely is an operation that, similar to the naïve computation of the determinant, is cubic in time complexity. To alleviate these two problems we need to make certain assumptions on the structure of the matrix W and indeed, many different types of linear transformations have appeared in the literature based on exactly these assumptions, e.g. W being a permutation matrix, i.e. a volume preserving (meaning it has a Jacobian determinant of 1) binary matrix with exactly one element per row being 1 and all other elements in the row being 0. Another, arguably more useful strategy, at least for high dimensional and/or complex data is to parameterize W through matrix decomposition, that is to say a product of matrices with specified structure, such that relatively cheap Jacobian determinant computation and inversion are guaranteed. Below we will briefly discuss one such popular decomposition.

PLU linear transform A matrix $W \in \mathbb{R}^{D \times D}$ can be written as a product of three matrices $P, L, U \in \mathbb{R}^{D \times D}$, where P is a permutation matrix, L is a lower triangular matrix and U is an upper triangular matrix:

$$W = PLU \tag{1.19}$$

By restricting the diagonal elements of L and U to be positive we can ensure that W is invertible. The determinant of W is simply the product of the diagonal elements of L and U:

$$\det W = \prod_{i}^{D} L_{ii} U_{ii} \tag{1.20}$$

Computing the inverse of W involves three steps:

1. Reversing the permutation
2. Solving an upper triangular system
3. Solving a lower triangular system

Solving the systems is an $\mathcal{O}(D^2)$ operation, thus computing the inverse of a PLU transformation is also an $\mathcal{O}(D^2)$ operation.

1.3.3.2 Coupling layers

Coupling layers are transformations of the following form. Given $\boldsymbol{x} \in \mathbb{R}^D$, $g(\cdot)$ the coupling layer transformation, $\boldsymbol{z} = g(\boldsymbol{x})$ and $d < D$:

$$\boldsymbol{z}_{1:d} = \boldsymbol{x}_{1:d} \tag{1.21}$$

$$\boldsymbol{z}_{d+1:D} = \boldsymbol{x}_{1:d} \odot \exp(s(\boldsymbol{x}_{1:d})) + t(\boldsymbol{x}_{1:d}) \tag{1.22}$$

where $s(\cdot), t(\cdot)$ are functions represented by neural networks and $\odot$ denotes the elementwise/Hadamard product.

The transformation described by eqs. 1.21 and 1.22 is quite simple and in practice, many of these layers are either composed in an alternating fashion or with invertible linear layers interspersed in-between to achieve a meaningful transformation of the data. In essence, coupling layers establish a dependence between parts of the input vector while ensuring that the computation of the Jacobian determinant remains tractable. The Jacobian of a coupling layer has the following form:

$$J_g(\boldsymbol{x}) = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial \boldsymbol{z}_{d+1:D}}{\partial \boldsymbol{x}_{1:d}} & \operatorname{diag}(\exp(s(\boldsymbol{x}_{1:d}))) \end{bmatrix} \tag{1.23}$$

where $\operatorname{diag}(\exp(s(\boldsymbol{x}_{1:d})))$ denotes the diagonal matrix, the diagonal elements of which are given by the vector $\exp(s(\boldsymbol{x}_{1:d}))$. The Jacobian of g is triangular, as such its determinant is simply the product of its diagonal elements. In this particular case the determinant is computed as $\exp\left(\sum_i^d s(\boldsymbol{x}_i)\right)$. We note that to compute the Jacobian of g, we do not need to compute the Jacobians of s and t, so the neural networks representing those functions can be arbitrarily complex. Inverting a coupling layer is trivial:

$$\boldsymbol{x}_{1:d} = \boldsymbol{z}_{1:d} \tag{1.24}$$

$$\boldsymbol{x}_{d+1:D} = (\boldsymbol{z}_{d+1:D} - t(\boldsymbol{z}_{1:d})) \odot \exp(-s(\boldsymbol{z}_{1:d})) \tag{1.25}$$

Evaluating the forward and inverse transformations carries the same computational complexity, in other words inference and sampling are made possible by paying the same price computationally.

1.3.3.3 Spline flow layers

As stated in the preceding section, coupling transformations are rather simplistic and one typically needs to compose many of them to achieve a flexible transformation of

the data. An alternative transformation is based on monotonic splines. Splines are functions that are defined piecewise in an interval $[A, B]$. This interval is divided in K segments and the overall spline transformation consists of simple functions defined in each segment, where subsequent segments agree on the knot points between them. Because the spline transformations are monotonically increasing, analytical invertibility is ensured. Spline transformations are applied elementwise in the input vector. Various spline flows based on monotonic polynomials have been proposed in the literature [Dolatabadi et al., 2020, Durkan et al., 2019a,b, Müller et al., 2019]. Below we will briefly discuss a flexible spline flow which is used predominantly in chapter 4.

Monotonic rational quadratic splines Durkan et al. [2019a] construct a rational quadratic spline flow in the interval $[-B, B]$ by defining rational-quadratic splines in K segments of the interval, the boundaries of which are given by $K+1$ monotonically increasing knot points $\{x^{(k)}, y^{(k)}\}_{k=0}^{K}$, where $\{x^{(0)}, y^{(0)}\} = (-B, -B)$ and $\{x^{(K)}, y^{(K)}\} = (B, B)$. To avoid discontinuities in the derivative of the transformation, and thus subsequent numerical instability during training, the derivatives in the internal $K-1$ knot points are set arbitrarily to $\{\delta^{(k)}, \delta^{(k)}\}_{k=1}^{K-1}$ positive values. Given $s_k = \frac{y^{k+1}-y^k}{x^{k+1}-x^k}$ and $\xi(x) = \frac{x-x^k}{x^{k+1}-x^k}$, a rational quadratic function $\frac{\alpha^{(k)}(\xi)}{\beta^{(k)}(\xi)}$ defined in segment k has the following form:

$$\frac{\alpha^{(k)}(\xi)}{\beta^{(k)}(\xi)} = y^{(k)} + \frac{(y^{(k+1)} - y^{(k)})[s^{(k)}\xi^2 + \delta^{(k)}\xi(1-\xi)]}{s^{(k)} + [\delta^{(k+1)} + \delta^{(k)} - 2s^{(k)}]\xi(1-\xi)} \tag{1.26}$$

The transformation is applied elementwise on the input vector, thus the logarithm of the absolute value of the Jacobian determinant is given by the sum of the log derivatives of eq. 1.26 with respect to each element x of the input vector $\boldsymbol{x}$:

$$\frac{d}{dx}\left[\frac{\alpha^{(k)}(\xi)}{\beta^{(k)}(\xi)}\right] = \frac{(s^{(k)})^2[\delta^{(k+1)}\xi^2 + 2s^{(k)}\xi(1-\xi) + \delta^{(k)}(1-\xi)^2]}{[s^{(k)} + [\delta^{(k+1)} + \delta^{(k)} - 2s^{(k)}]\xi(1-\xi)]^2} \tag{1.27}$$

To evaluate the inverse of the rational quadratic function we need to solve for the roots of a quadratic equation and since the equation is monotonic we can always determine the appropriate root. For more details on the relevant expressions and procedure please see Durkan et al. [2019a].

1.4 Variational Autoencoders

1.4.1 Latent variable models

Latent variable models are parametric models of the following form:

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}, \tag{1.28}$$

where $\boldsymbol{x}$ are considered the *observed variables* and $\boldsymbol{z}$ the *hidden* or *latent variables*. The joint distribution $p_\theta(\boldsymbol{x}, \boldsymbol{z})$ is called the *complete data likelihood*. Typically, the joint distribution factorizes in the following way:

$$p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}), \tag{1.29}$$

where the conditional distribution over the observations $\boldsymbol{x}$ is called the *likelihood* and $p(\boldsymbol{z})$ is the *prior distribution* over latent variables $\boldsymbol{z}$. The decomposition in eq. 1.29 is prevalent in generative modeling, where it is used to model the generative process of observations $\boldsymbol{x}$ in the following sense: a latent variable representing some defining attribute (e.g. the angle of rotation for an object in the case of image data or the class of the observation in the case of discrete latent variables) of observation $\boldsymbol{x}$ is first sampled from the prior $p(\boldsymbol{z})$, which is then used to sample an observed variable from the conditional distribution $p_\theta(\boldsymbol{x}|\boldsymbol{z})$.

During inference we are interested in computing the posterior distribution over the latent variables:

$$p_\theta(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{x})} = \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{\int p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}} \tag{1.30}$$

However, in practice the integral in eq. 1.28 is often intractable (e.g. due to a high number of dimensions), in which case we need to resort to approximate methods. At this point we have two options: either resort to sampling methods like Markov Chain Monte Carlo (MCMC) to approximately draw samples from the true posterior or view the approximation of the posterior as an optimization problem, where we need to optimize the parameters of a parametric model to approximate the true posterior. The latter approach is generally known as *variational inference*.

1.4.2 Variational inference

MCMC constructs an ergodic Markov chain that has $p(\boldsymbol{z}|\boldsymbol{x})$ as its stationary distribution. The sampler then yields samples from the chain/stationary distribution. If we want to evaluate the posterior we compute an empirical estimate from these samples.

MCMC comes with guarantees for asymptotically yielding exact samples from the target distribution [Blei et al., 2017, Robert et al., 1999], however it is not suited to high dimensional datasets or complicated models due to the fact that it generally tends to be a computationally intensive method. Thus, in terms of computational expediency variational inference is a more practical choice, although it comes at the cost of the (asymptotic) guarantees MCMC carries. In this thesis, posterior approximation is relevant for chapters 2 and 3, where between MCMC and variational inference we opted for the latter method.

As stated in the previous section variational inference turns the inference problem into an optimization one. In the current section we will explain the overall setting and the objective function of the approach. We are interested in approximating the true posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$. To do so, we first specify a *variational family* of distributions parameterized by ϕ. We will optimize ϕ to arrive at the member of this family which best approximates $p(\boldsymbol{z}|\boldsymbol{x})$ and we will measure the quality of this approximation via the KL divergence. We can now formulate our problem:

$$q^\star(\boldsymbol{z}) = \arg\min_{\phi} D_{\mathrm{KL}}(q_\phi(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})), \tag{1.31}$$

Solving this minimization problem gives us the best approximating distribution within a given variational family. Recall, however, that we cannot compute this KL divergence, since we cannot compute $p(\boldsymbol{z}|\boldsymbol{x})$:

$$D_{\mathrm{KL}}(q_\phi(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})) = \mathbb{E}_q[\log q_\phi(\boldsymbol{z})] - \mathbb{E}_q[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] + \log p(\boldsymbol{x}) \tag{1.32}$$

We can optimize the alternative objective:

$$\mathcal{L}(\boldsymbol{x}, \theta, \phi) = \mathbb{E}_q[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] - \mathbb{E}_q[q_\phi(\boldsymbol{z})] \tag{1.33}$$

This function provides a lower bound to the marginal likelihood $\log p(\boldsymbol{x})$. The KL divergence is a non-negative quantity and it is equal to zero when the distributions match and so the tightness of the bound depends on the quality of the approximation of the posterior. Thus, rearranging eq. 1.32 we get:

$$\log p(\boldsymbol{x}) = \mathbb{E}_q[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] - \mathbb{E}_q[q_\phi(\boldsymbol{z})] + D_{\mathrm{KL}}(q_\phi(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})) \tag{1.34}$$

$$= \mathcal{L}(\boldsymbol{x}, \theta, \phi) + D_{\mathrm{KL}}(q_\phi(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})), \tag{1.35}$$

which given what we stated above, we can express as a lower bound, with equality achieved when the term $D_{\mathrm{KL}}(q_\phi(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x}))$ is zero:

$$\log p(\boldsymbol{x}) \geq \mathcal{L}(\boldsymbol{x}, \theta, \phi) \tag{1.36}$$

Thus we can reframe our optimization problem expressed in eq. 1.31 as one where we equivalently want to maximize this lower bound.

The marginal log-likelihood $\log p(\boldsymbol{x})$ is also called the *evidence* and thus the function $\mathcal{L}(\boldsymbol{x}, \theta, \phi)$ is called the *evidence lower bound* (ELBO). Often, the ELBO is expressed

in terms of the conditional log-likelihood and the KL divergence between the approximate posterior $q_\phi(\boldsymbol{z})$ and the prior $p(\boldsymbol{z})$:

$$\mathcal{L}(\boldsymbol{x}, \theta, \phi) = \mathbb{E}_q[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] + \mathbb{E}_q[p(\boldsymbol{z})] \tag{1.37}$$
$$= \mathbb{E}_q[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] + D_{\mathrm{KL}}(q_\phi(\boldsymbol{z})||p(\boldsymbol{z})) \tag{1.38}$$

1.4.3 Autoencoding variational inference

Historically, the approach employed in variational inference is based on the following choice for the variational family:

$$q(\boldsymbol{z}) = \prod_i q_i(z_i), \tag{1.39}$$

in which latent variables are independent and each of them is governed by the density q_i. This implies that each data point $\boldsymbol{x}$ is governed by its own variational parameters. We note that this family is *not* a function of data points $\boldsymbol{x}$ and the only way in which they are connected to the conditional likelihood is through the optimization problem [Blei et al., 2017]. This approach is called *mean-field variational inference*.

In recent years, however, a different approach has gained traction, which is based on *amortized inference* [Gershman and Goodman, 2014]. The focus of amortized inference is the sharing of variational parameters across data points, as opposed to the mean-field method which optimizes variational parameters separately for each data point. The paradigm which best exemplifies amortized inference with respect to posterior inference is a neural network-based approach called *Variational autoencoders* (VAEs) [Kingma and Welling, 2014, Rezende et al., 2014]. VAEs jointly optimize the parameters of the variational family ϕ and the likelihood θ. In VAEs the variational family is actually a function of data points $\boldsymbol{x}$, such that we can rewrite it as $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and is often called the *inference/recognition model* or *encoder*, while the conditional likelihood $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is called the *decoder*. Both of these densities are parameterized by neural networks. A typical choice for the encoder is a Gaussian distribution with diagonal covariance:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\mu_\phi(\boldsymbol{x}), \mathrm{diag}(\sigma^2_\phi(\boldsymbol{x}))), \tag{1.40}$$

where parameter vectors μ and σ^2 are output by a neural network with parameters ϕ. The prior density is usually defined as a standard Gaussian. Finally, the likelihood/decoder density is chosen according to the data modality, e.g. a Bernoulli distribution would be the appropriate choice for binary data, while a Gaussian distribution would be better suited to real valued data, etc. From an autoencoder point of view, the likelihood term in eq. 1.38 can be viewed as a reconstruction term while the KL term can be viewed as a regularizer.

To evaluate the expectations in eq. 1.38 we need samples from the approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. This presents us with the problem of having to backpropagate through a sampling step during training. Kingma and Welling [2014] showed that for certain distributions, sampling can be re-expressed as a deterministic transformation of a random variable, which is differentiable with respect to its parameters. For Gaussian distributions this amounts to sampling a random variable $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ and then applying the transformation $\boldsymbol{z} = \mu_\phi(\boldsymbol{x}) + \sigma_\phi(\boldsymbol{x}) \odot \boldsymbol{\epsilon}$, where $\mu_\phi(\boldsymbol{x}), \sigma_\phi(\boldsymbol{x})$ are respectively the mean and variance parameters of the Gaussian distribution, parameterized by neural networks with parameters ϕ. By deriving samples in this way, we are able to backpropagate through deterministic functions μ_ϕ and σ_ϕ. In the context of VAEs this is called the *reparameterization trick* and it allows for low variance estimates of gradients with respect to parameters.

CHAPTER 2

Variational Autoencoders with Riemannian Brownian Motion Priors

This chapter is adapted from Kalatzis et al. [2020]

Authors: Dimitris Kalatzis, David Eklund, Georgios Arvanitidis, Søren Hauberg

Abstract

Variational Autoencoders (VAEs) represent the given data in a low dimensional latent space, which is generally assumed to be Euclidean. This assumption naturally leads to the common choice of a standard Gaussian prior over continuous latent variables. Recent work has, however, shown that this prior has a detrimental effect on model capacity, leading to subpar performance. We propose that the Euclidean assumption lies at the heart of this failure mode. To counter this, we assume a Riemannian structure over the latent space, which constitutes a more principled geometric view of the latent codes, and replace the standard Gaussian prior with a Riemannian Brownian motion prior. We propose an efficient inference scheme that does not rely on the unknown normalizing factor of this prior. Finally, we demonstrate that this prior significantly increases model capacity using only one additional scalar parameter.

2.1 Introduction

Variational autoencoders (VAEs) [Kingma and Welling, 2014, Rezende et al., 2014] simultaneously learn a conditional density $p(\boldsymbol{x}|\boldsymbol{z})$ of high dimensional observations

and low dimensional representations $\boldsymbol{z}$ giving rise to these observations. In VAEs, a prior distribution $p(\boldsymbol{z})$ is assigned to the latent variables which is typically a standard Gaussian. It has, unfortunately, turned out that this choice of distribution is limiting the modelling capacity of VAEs and richer priors have been proposed instead [Bauer and Mnih, 2018, Klushyn et al., 2019, Tomczak and Welling, 2017, van den Oord et al., 2017]. In contrast to this popular view, we will argue that the limitations of the prior are not due to *lack of capacity*, but rather *lack of principle*.

Informally, the Gaussian prior has two key problems.

1. The Euclidean representation is arbitrary. Behind the Gaussian prior lies the assumption that the latent space $\mathcal{Z}$ is Euclidean. However, if the decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is of sufficiently high capacity, then it is always possible to reparameterize the latent space from $\boldsymbol{z}$ to $h(\boldsymbol{z}), h : \mathcal{Z} \to \mathcal{Z}$, and then let the decoder invert this reparameterization as part of its decoding process [Arvanitidis et al., 2018, Hauberg, 2018b]. This implies that we cannot assign any meaning to specific instantiations of the latent variables, and that Euclidean distances carry limited meaning in $\mathcal{Z}$. This is an *identifiability problem* and it is well-known that even the most elementary latent variable models are subject to such. For example, Gaussian mixtures can be reparameterized by permuting cluster indices, and principal components can be arbitrarily rotated [Bishop, 2006].

2. Latent manifolds are mismapped onto $\mathcal{Z}$. In all but the simplest cases, the latent manifold $\mathcal{M}$ giving rise to data observations is embedded in $\mathcal{Z}$. An encoder with adequate capacity will always recover some smoothened form of $\mathcal{M}$, which will either result in the latent space containing "holes" of low density or, in $\mathcal{M}$ being mapped to the whole of $\mathcal{Z}$ under the influence of the prior. Both cases will lead to bad samples or convergence problems. This problem is called *manifold mismatch* [Davidson et al., 2018a, Falorsi et al., 2018] and is closely related to *distribution mismatch* [Bauer and Mnih, 2018, Hoffman and Johnson, 2016, Rosca et al., 2018] where the prior samples from regions to which the variational posterior (or encoder) does not assign any density. A graphical illustration of this situation can be seen on the left panel of Fig. 2.1, where a VAE is trained on the 1-digits of MNIST under the

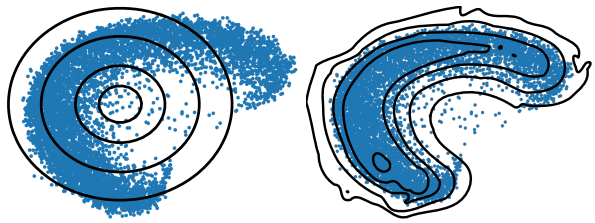


Figure 2.1. The latent space priors of two VAEs trained on the digit *1* from MNIST. *Left:* Using a unit Gaussian prior. *Right:* Using a Riemannian Brownian motion (ours) with trainable (scalar) variance.

Gaussian prior. The prior assigns density where there is none.

In this paper, we consider an alternative prior, which is shown in the right panel of Fig. 2.1. This is a Riemannian Brownian motion model defined over the manifold immersed by the decoder. The Riemannian structure solves the identifiability problem and gives a meaningful representation that is invariant to reparametrizations and at the same time restricts the prior to sample only from the image of $\mathcal{M}$ onto $\mathcal{Z}$. The prior generalizes the Gaussian to the Riemannian setting. It only has a single scalar variance parameter, yet it is able to capture intrinsic complexities in the data.

2.2 Background

2.2.1 Variational autoencoders

VAEs learn a generative model $p_\theta(\boldsymbol{x}, \boldsymbol{z})$ by specifying a likelihood of observations conditioned on latent variables $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ and a prior over the latent variables $p(\boldsymbol{z})$. The marginal likelihood of the observations $p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})\mathrm{d}\boldsymbol{z}$ is intractable. As such, VAEs are trained by maximizing the variational *Evidence Lower Bound* (ELBO) on the marginal likelihood :

$$\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})), \tag{2.1}$$

where $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ denotes the variational family. Kingma and Welling [2014], Rezende et al. [2014] proposed a low variance estimator of stochastic gradients of the ELBO, known as *reparameterization trick*.

In the VAE framework, both the variational family $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the conditional likelihood $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ are parameterized by neural networks with variational parameters ϕ and generative parameters θ. In the language of autoencoders, these networks are often called *encoder* and *decoder* parameterizing the variational family and the generative model respectively. From an autoencoder perspective, Eq. 2.1 can be seen as a loss function involving a data reconstruction term (the generative model) and a regularization term (the KL divergence between the variational family and the prior distribution over the latent variables).

2.2.2 VAE decoders as immersions

We will dedicate this subsection to showing that, under certain architectural choices, VAE decoders induce Riemannian metrics in the latent space. That is to say, they belong to a certain class of maps, called *smooth immersions*, which give rise to *immersed submanifolds*. In other words, we will formally describe our intuition about

VAEs mapping the latent space back to data space, using the language of smooth manifolds and Riemannian geometry.

The generative and variational distributions can be seen as families of parameterized mappings $g_\phi : \mathcal{X} \to \mathcal{Z}$ and $f_\theta : \mathcal{Z} \to \mathbb{R}^M$, $\mathcal{Z} \subset \mathbb{R}^N$ and $M > N$ and parameters ϕ and θ respectively. The family defined by the generative model is of particular interest. To make the subsequent exposition clearer we will assume a Gaussian generative model and rewrite it in the following form:

$$f_\theta(\mathbf{z}) = \mu_\theta(\mathbf{z}) + \sigma_\theta(\mathbf{z}) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_M) \tag{2.2}$$

with $\mu_\theta : \mathcal{Z} \to \mathbb{R}^M$, $\sigma_\theta : \mathcal{Z} \to \mathbb{R}^M_+$, denoting the mean and standard deviation of the generative model parameterized by neural networks with parameters θ and $\odot$ denoting the Hadamard or element-wise product.

Definition 5. ***(Smooth immersions)*** *Given smooth manifolds $\mathcal{M}$ and $\mathcal{M}'$ with $dim(\mathcal{M}) < dim(\mathcal{M}')$, a mapping $f : \mathcal{M} \to \mathcal{M}'$, a point $p \in \mathcal{M}$ and its image $f(p) \in \mathcal{M}'$, the mapping f is called an immersion if its differential $df_p : T_p\mathcal{M} \to T_{f(p)}\mathcal{M}'$ is injective for all $p \in \mathcal{M}$.*

We will consider a particular Riemannian metric on $\mathcal{Z}$ induced by μ_θ and σ_θ. The architectures of μ_θ and σ_θ are such that these maps are immersions. Consider now the *diagonal immersion*

$$f : \mathcal{Z} \to \mathbb{R}^M \times \mathbb{R}^M_+ : \boldsymbol{z} \mapsto (\mu_\theta(\boldsymbol{z}), \sigma_\theta(\boldsymbol{z})), \tag{2.3}$$

whose geometry encodes both mean and variance. The random map f_θ is a random projection given by ϵ of the diagonal immersion. Sampling using the decoder can therefore be seen as first sampling the image of this immersion and then randomly projecting down to $\mathcal{X}$ [Eklund and Hauberg, 2019]. Taking the pull-back metric $\mathbf{G}_\mathbf{z}$ of f to $\mathcal{Z}$ we obtain

$$\mathbf{G}_\mathbf{z} = J_{\boldsymbol{\mu}}(\boldsymbol{z})^\top J_{\boldsymbol{\mu}}(\boldsymbol{z}) + J_{\boldsymbol{\sigma}}(\boldsymbol{z})^\top J_{\boldsymbol{\sigma}}(\boldsymbol{z}), \tag{2.4}$$

where $J_{\boldsymbol{\mu}}$ and $J_{\boldsymbol{\sigma}}$ are the Jacobian matrices of μ_θ and σ_θ.

The metric $\mathbf{G}_\mathbf{z}$ was studied by Arvanitidis et al. [2018] and is known to yield geodesics that follow high density regions in latent space. As an example, Fig. 2.2 shows geodesics of a VAE trained on 1-digits from MNIST, which follow the data due to the variance term of the metric, which penalizes geodesics going through low density regions of the latent space.

2.3 Geometric latent priors

The standard Gaussian prior relies on the usual Lebesgue measure which in turn, assumes a Euclidean structure over the latent space $\mathcal{Z}$. Recently, it has been noted

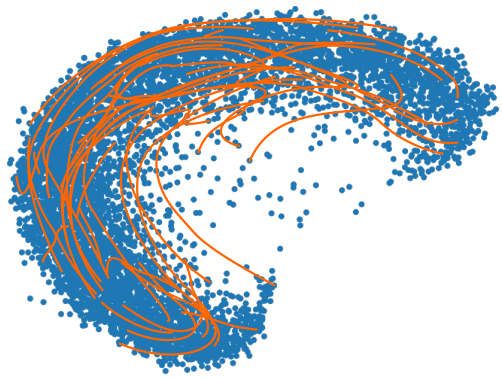


Figure 2.2. Example geodesics under the pull-back metric equation 2.4. The associated VAE is the same as in Fig. 2.1.

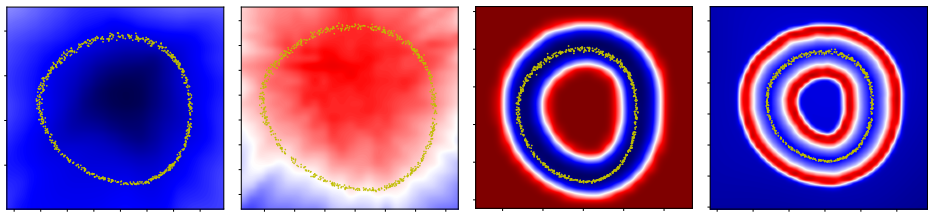


Figure 2.3. Inferred latent space for a toy data set, embedded via a non-linear function in $\mathbb{R}^{100}$. The background color, with blue representing lower and red representing higher values, from left to right, show: the (log) standard deviation estimated by a typical neural network; the associated (log) volume measure; the RBF (log) standard deviation estimate; and the associated (log) volume measure. Best viewed in color.

[Arvanitidis et al., 2018, Hauberg, 2018b] that this assumption is mathematically questionable, and that, empirically, Euclidean latent space distances carry little information about the relationship between data points. Rather, a Riemannian interpretation of the latent space appears more promising. It is evident that the geometric structure over the latent space carries significant information about data density that the traditional Euclidean interpretation foregoes. With this in mind, we propose that the prior should be defined with respect to the geometric structure. We could opt for a *Riemannian normal distribution*, which is well-studied [Arvanitidis et al., 2016a, Hauberg, 2018a, Mardia and Jupp, 2000, Oller, 1993, Pennec, 2006a]. Unfortunately, computing its normalization constant is expensive and involves Monte Carlo integration. Furthermore, it is equally hard to sample from this distribution, since it generally requires rejection sampling with non-trivial proposal distributions.

Instead we consider a cheap and flexible alternative, namely the heat kernel of a Brownian motion process [Hsu, 2002]. A Brownian motion X_t on an immersed Riemannian manifold $\mathcal{M} \subseteq \mathbb{R}^M$ can be defined through a stochastic differential equation

on Stratonovich form:

$$\mathrm{d}X_t = \sum_{\alpha=1}^{M} P_\alpha(X_t) \circ \mathrm{d}W_t^\alpha. \tag{2.5}$$

Here $W_t = (W_t^1, \ldots, W_t^M)$ is a Brownian motion in $\mathbb{R}^M$ and $P_1(X_t), \ldots, P_M(X_t)$ denotes the projection of the standard basis of $\mathbb{R}^M$ onto the tangent space of $\mathcal{M}$ at X_t. This way, a Brownian motion on $\mathcal{M}$ is driven by a Euclidean Brownian motion W_t projected to the tangent space. Fixing an initial point $\boldsymbol{\mu} \in \mathcal{M}$ and a time $t > 0$, Brownian motion starting at $\boldsymbol{\mu}$ running for time t gives rise to a random variable on $\mathcal{M}$. Its density function is the *transition density* $p(\boldsymbol{x})$. An alternative description of Brownian motion on $\mathcal{M}$ is that $p(\boldsymbol{x})$ is the heat kernel associated to the Laplace-Beltrami operator of a scalar function h on $\mathcal{M}$:

$$\Delta h = d\mathcal{M}^{-1} \partial_i \left(d\mathcal{M} g^{ij} \partial_j h \right) \tag{2.6}$$

where $d\mathcal{M}$ is the volume measure of the immersed submanifold $\mathcal{M}$, g^{ij} are the components of the inverse metric tensor and $\partial_i := \frac{\partial}{\partial x^i}, \partial_j := \frac{\partial}{\partial x^j}$ are the basis vectors at the tangent space $T_p\mathcal{M}$. We will express the transition density in terms of local coordinates $\mathcal{Z} \to \mathcal{M}$ on $\mathcal{M}$. Conveniently, we may approximate the transition density by a so-called Parametrix expansion in a power series [Hsu, 2002]. In this paper we will use the zeroth order approximation which gives rise to the following expression for $p(\boldsymbol{z})$ with $\boldsymbol{z} \in \mathcal{Z}$:

$$p(\boldsymbol{z}) \approx (2\pi t)^{-d/2} H_0 \exp\left(-\frac{l^2(\boldsymbol{z}, \boldsymbol{\mu})}{2t} \right), \tag{2.7}$$

where:

- $t \in \mathbb{R}$, denotes the duration of the Brownian motion, and corresponds to *variance* on Euclidean manifolds.
- d is the dimensionality of $\boldsymbol{z}$.
- $\boldsymbol{\mu} \in \mathcal{Z}$ is the center of the Brownian motion.
- $l(\cdot, \cdot)$ is the geodesic distance on the manifold.
- $H_0 = (\frac{\det G_{\boldsymbol{z}}}{\det G_{\boldsymbol{\mu}}})^{1/2}$ is the ratio of the Riemannian volume measure evaluated at points $\boldsymbol{z}$ and $\boldsymbol{\mu}$ respectively.

Equation 2.7 can be evaluated reasonably fast as no Monte Carlo integration is required. The most expensive computation is the evaluation of the geodesic distance for which several efficient algorithms exist [Arvanitidis et al., 2019, Hennig and Hauberg, 2014]. Here we parameterize the geodesic as a cubic spline and perform direct energy minimization.

2.3.1 Inference

Since we use the heat kernel density function for the prior $p(\boldsymbol{z})$, we need the variational family $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to be defined with respect to the same Riemannian measure. We therefore also use the heat kernel density function for the variational family, which is parameterized by the encoder network with variational parameters ϕ. The parameter t of the prior is learned through optimization. The ELBO can be derived with respect to the volume measure $d\mathcal{M}$:

$$\begin{aligned}
\log & p(\boldsymbol{x}) \geq \mathcal{L}_{\mathcal{M}}(\boldsymbol{x};\theta,\phi) \\
&\triangleq \int_{\mathcal{M}} \log\left(\frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right) q_\phi(\boldsymbol{z}|\boldsymbol{x}) d\mathcal{M}_{\boldsymbol{z}} \\
&= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})).
\end{aligned} \tag{2.8}$$

This ELBO can be estimated using Monte Carlo samples from the variational posterior. With no analytical solution to the KL divergence we resort to Monte Carlo integration:

$$\begin{aligned}
D_{\mathrm{KL}}(q||p) &= \int_{\mathcal{M}} \log\frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})} q_\phi(\boldsymbol{z}|\boldsymbol{x}) d\mathcal{M}_{\boldsymbol{z}} \\
&= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log q(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{z})] \\
&\approx \frac{1}{N}\sum_{i=1}^{N}(\log q(\boldsymbol{z}_i|\boldsymbol{x}) - \log p(\boldsymbol{z}_i))
\end{aligned} \tag{2.9}$$

with:

$$\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) = -\frac{d}{2}\log(2\pi t_q) + \log H_{0,q} - \frac{l_q^2}{2t_q} \tag{2.10}$$

$$\log p(\boldsymbol{z}) = -\frac{d}{2}\log(2\pi t_p) + \log H_{0,p} - \frac{l_p^2}{2t_p} \tag{2.11}$$

where $l_q^2 = l^2(\boldsymbol{z}, \boldsymbol{\mu}_q)$, $l_p^2 = l^2(\boldsymbol{z}, \boldsymbol{\mu}_p)$.

Thus, the final form of the Monte Carlo evaluation of the KL divergence is:

$$\begin{aligned}
D_{\mathrm{KL}}(q||p) \approx \frac{1}{2}\Bigg[\frac{1}{N}\sum_{i=1}^{N}\Bigg(&\log\det G_{\mu_p}(\boldsymbol{z}_i) - \\
&\log\det G_{\mu_q}(\boldsymbol{z}_i) + \frac{l^2(\boldsymbol{z}_i,\boldsymbol{\mu}_p)}{t_p} - \frac{l^2(\boldsymbol{z}_i,\boldsymbol{\mu}_q)}{t_q}\Bigg) \\
&+ d(\log t_p - \log t_q)\Bigg]
\end{aligned} \tag{2.12}$$

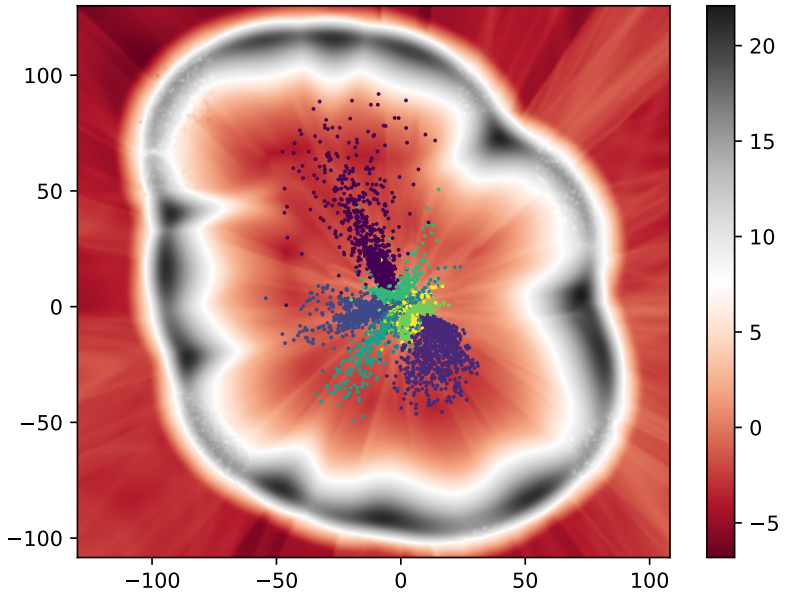


Figure 2.4. Latent space of an $\mathcal{R}$-VAE, plotted against the Riemannian volume measure $d\mathcal{M}$. Once again note the "borders" created by the metric roughly demarcating the latent code support. The latent codes are colored according to label. Best viewed in color.

2.3.2 Sampling

In the previous section we mentioned that a Brownian motion (BM) on the manifold can be derived by projecting each BM step X_t on the tangent space at t. However we will take each step directly in the latent space and avoid having to evaluate the exponential map. Given a manifold $\mathcal{M}$ with dimension N, the immersion $f : \mathcal{M} \to \mathbb{R}^M$, a point $\mathbf{a} \in \mathcal{M}$ and its image under f, $\boldsymbol{A} \in \mathbb{R}^M$ we take a random step from $\boldsymbol{A}$:

$$\Delta \sim \mathcal{N}(\mathbf{0}, \Sigma_M). \tag{2.13}$$

Applying a Taylor expansion we have:

$$f(\mathbf{a} + \epsilon) = f(\mathbf{a}) + \mathbf{J}_\mathbf{a}\epsilon + \mathcal{O}\left(\epsilon^2\right). \tag{2.14}$$

With $\Delta = f(\mathbf{a} + \epsilon) - f(\mathbf{a})$ we have:

$$\Delta = \mathbf{J}_\mathbf{a}\epsilon + \mathcal{O}\left(\epsilon^2\right). \tag{2.15}$$

For small ϵ an approximation to taking a step directly in the latent space is then $\mathbf{b} = \mathbf{a} + \epsilon$ with $\epsilon \approx \mathbf{J}_\mathbf{a}^+\Delta$ and $\mathbf{J}_\mathbf{a}^+ = \left(\mathbf{J}_\mathbf{a}^\top \mathbf{J}_\mathbf{a}\right)^{-1}\mathbf{J}_\mathbf{a}^\top \in \mathbb{R}^{N\times M}$ the pseudoinverse of $\mathbf{J}_\mathbf{a}$.

Since $\Delta \sim \mathcal{N}(\mathbf{0}, \Sigma_M)$ the step ϵ can be written:

$$\epsilon \sim \mathcal{N}\left(\mathbf{0}, \mathbf{J}_{\mathbf{a}}^{+} \Sigma_M \left(\mathbf{J}_{\mathbf{a}}^{+}\right)^{\top}\right). \tag{2.16}$$

We consider an isotropic heat kernel so in our case $\Sigma_M = \sigma^2 \mathbf{I}$. Furthermore:

$$\begin{aligned}
\mathbf{J}_{\mathbf{a}}^{+} \Sigma_M \left(\mathbf{J}_{\mathbf{a}}^{+}\right)^{\top} &= \left(\mathbf{J}_{\mathbf{a}}^{\top} \mathbf{J}_{\mathbf{a}}\right)^{-1} \mathbf{J}_{\mathbf{a}}^{\top} \Sigma_M \mathbf{J}_{\mathbf{a}} \left(\mathbf{J}_{\mathbf{a}}^{\top} \mathbf{J}_{\mathbf{a}}\right)^{-\top} \\
&= \sigma^2 \left(\mathbf{J}_{\mathbf{a}}^{\top} \mathbf{J}_{\mathbf{a}}\right)^{-1} \mathbf{J}_{\mathbf{a}}^{\top} \mathbf{J}_{\mathbf{a}} \left(\mathbf{J}_{\mathbf{a}}^{\top} \mathbf{J}_{\mathbf{a}}\right)^{-\top} \\
&= \sigma^2 \left(\mathbf{J}_{\mathbf{a}}^{\top} \mathbf{J}_{\mathbf{a}}\right)^{-\top} = \sigma^2 \left(\mathbf{J}_{\mathbf{a}}^{\top} \mathbf{J}_{\mathbf{a}}\right)^{-1}.
\end{aligned} \tag{2.17}$$

This implies that

$$\epsilon \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \left(\mathbf{J}_{\mathbf{a}}^{\top} \mathbf{J}_{\mathbf{a}}\right)^{-1}\right). \tag{2.18}$$

Thus, to sample from the prior we simply need to run Brownian motion for $t = 1, \ldots, T$:

$$\boldsymbol{z}_t \sim \mathcal{N}\left(\boldsymbol{z}_{t-1}, \frac{\sigma^2}{T} \left(J_{\boldsymbol{z}_{t-1}}^{\top} J_{\boldsymbol{z}_{t-1}}\right)^{-1}\right) \tag{2.19}$$

An obvious concern regarding the computational cost of sampling is the inverting of the metric tensor. While this is a valid concern for large latent dimensionalities, in practice and for the typical number of latent dimensions found in generative modelling literature the sampling cost is bearable, considering that the operation can be parallelized for K samples. We further note that from a practical standpoint for small diffusion times the number of discretized steps can be small. The time complexity of the sampling operation is

$$\mathcal{O}(KHM + KMN^2 + N^3) \tag{2.20}$$

where K is the number of samples, N is the latent space dimensionality, M is the input space dimensionality and H is the decoder hidden layer size.

2.4 Meaningful variance estimation

We now turn to the problem of restricting our prior to sample from the image of our manifold in $\mathcal{Z}$. Since typically the geometry of the data is not known a priori, we adopt the Bayesian approach and relate uncertainty estimation in the generative model to the geometry of the latent manifold. Specifically, since the generative model

parameterizes $f_\theta : \mathcal{Z} \to \mathcal{X}$ we construct it such that the pull-back metric will acquire high values away from the data support and thereby restrict prior samples to high density regions of the latent manifold.

In Sec. 2.2.2 we described the metric tensor arising from the diagonal immersion f. By the form of the metric, it is clear that both $\mu_\theta(\boldsymbol{z})$ and $\sigma_\theta(\boldsymbol{z})$ contribute to the manifold geometry. In recent works [Arvanitidis et al., 2018, Detlefsen et al., 2019a, Hauberg, 2018b] it was shown that neural network variance estimates are typically poor in regions away from the training data, due to poor extrapolation properties. Thus, neural networks cannot be trusted to properly estimate the variance of the generative model "off-the-shelf" when the functional form of the immersion (and thus the geometry of the data) is not known a priori. By extension, this leads to poor estimates of latent manifold geometry and latent densities. Arvanitidis et al. [2018] propose to use a *radial basis function (RBF)* network [Que and Belkin, 2016] to estimate precision, rather than variance. We adopt this approach due to its simplicity and relative numerical stability, however we note that similar approaches for principled variance estimation exist [Detlefsen et al., 2019a, Stirn and Knowles, 2020].

The influence of the RBF network can be seen in Fig. 2.3, where it is compared with a usual neural network variance estimate. Note that the metric creates "borders" demarcating the regions to which the latent codes have been mapped by the encoder. This makes interpolations and random walks generally follow the trend of the latent points instead of wondering off the support. Thus, this regularization scheme restricts prior sampling to such high density regions. A similar effect is not observed in the usual Gaussian VAE, where the prior samples from regions to which the variational posterior has not necessarily placed probability density [Hoffman and Johnson, 2016, Rosca et al., 2018].

2.5 Experiments

2.5.1 Generative modelling

For our first experiment we train a VAE with a Riemannian Brownian motion prior ($\mathcal{R}$-VAE) for different latent dimensions and compare it to a VAE with a standard Normal prior and a VAE with a VampPrior. Tables 2.1 & 2.2 show the results. $\mathcal{R}$-VAE achieves a better lower bound than both its Euclidean counterparts. The Brownian motion prior adapts to the latent code support and as such yields more expressive representations. On the other hand, with only a single parameter it results in a model that generalizes better than VAEs with a VampPrior.

Table 2.1. Results on MNIST (mean & std deviation over 10 runs). *Rec* denotes the negative conditional likelihood.

Model	Neg. ELBO	Rec	KL
VAE			
d = 2	-1030.38$_{\pm 5.34}$	-1033.06$_{\pm 5.48}$	2.68$_{\pm .14}$
d = 5	-1076.64$_{\pm 4.48}$	-1078.91$_{\pm 4.44}$	2.27$_{\pm .04}$
d = 10	-1110.79$_{\pm 1.17}$	-1113.01$_{\pm 1.13}$	2.22$_{\pm .03}$
VAE-VampPrior			
d = 2	-1045.03$_{\pm 5.22}$	-1047.34$_{\pm 5.20}$	2.30$_{\pm .03}$
d = 5	-1109.74$_{\pm 4.87}$	-1111.63$_{\pm 4.87}$	1.88$_{\pm .01}$
d = 10	-1116.58$_{\pm 4.23}$	-1118.27$_{\pm 4.20}$	1.69$_{\pm .02}$
$\mathcal{R}$-VAE			
d = 2	**-1047.29**$_{\pm 2.77}$	**-1053.70**$_{\pm 2.75}$	14.33$_{\pm .01}$
d = 5	**-1141.06**$_{\pm 7.09}$	**-1177.86**$_{\pm 3.39}$	28.00$_{\pm .2}$
d = 10	**-1170.03**$_{\pm 18.52}$	**-1280.94**$_{\pm 14.67}$	57.76$_{\pm 3.85}$

2.5.2 Classification

We next assess the usefulness of the latent representations of $\mathcal{R}$-VAE. Fig. 2.4 shows the latent code clusters. $\mathcal{R}$-VAE has produced more separable clusters in the latent space due to the prior adapting to the latent codes, which results in a less regularized clustering. We quantitatively measured the utility of the $\mathcal{R}$-VAE latent codes in different dimensionalities by training a classifier to predict digit labels and measuring the average overall and per-digit F1 score. Table 2.3 shows the results when comparing against the same classifier trained on latent codes derived by a VAE. $\mathcal{R}$-VAE has a significant advantage in low dimensions. As dimensionality increases this advantage becomes non-existent. An explanation for this is that due to the KL annealing of the Euclidean VAE, its representations have become more informative.

2.5.3 Qualitative results

Finally we explore the geometric properties of a $\mathcal{R}$-VAE with a 2-dimensional latent space. Fig 2.4 shows the learned manifold. As in Fig. 2.3, the influence of the variance network on the metric can be seen in the "borders" surrounding the latent code support.

We begin by investigating the behavior of distances on the induced manifold. Fig. 2.5 shows the geodesic curves between two pairs of random points on the manifold, com-

Table 2.2. Results on FashionMNIST (mean & std deviation over 10 runs). *Rec* denotes the negative conditional likelihood.

Model	Neg. ELBO	Rec	KL
VAE			
d = 2	-443.13$_{\pm 10.67}$	-447.44$_{\pm 10.8}$	4.31$_{\pm .14}$
d = 5	-511.65$_{\pm 3.70}$	-517.41$_{\pm 3.84}$	5.76$_{\pm .21}$
d = 10	-525.05$_{\pm 5.87}$	-530.86$_{\pm 5.9}$	5.81$_{\pm .05}$
VAE-VampPrior			
d = 2	-705.90$_{\pm 17.3}$	-708.45$_{\pm 17.29}$	2.54$_{\pm .01}$
d = 5	-769.27$_{\pm 5.}$	-770.1$_{\pm 5.02}$	0.83$_{\pm .09}$
d = 10	-774.17$_{\pm 10.83}$	-777.75 $_{\pm 10.78}$	3.57$_{\pm .06}$
$\mathcal{R}$-VAE			
d = 2	**-708.77**$_{\pm 6.93}$	**-722.41**$_{\pm 5.736}$	13.64$_{\pm 1.51}$
d = 5	**-889.62**$_{\pm 3.44}$	**-913.61**$_{\pm 3.38}$	23.83$_{\pm .8}$
d = 10	**-959.2**$_{\pm 5.37}$	**-1001.4**$_{\pm 4.08}$	40.35$_{\pm .8}$

Table 2.3. Per digit and average F1 score for a classifier trained on the learned latent codes of VAE and $\mathcal{R}$-VAE. Results are averaged over 5 classifier training runs.

Digits	0	1	2	3	4	5	6	7	8	9	Avg
VAE											
d = 2	0.94	0.95	0.88	0.67	0.55	0.42	0.86	0.68	0.61	0.53	0.72$_{\pm .002}$
d = 5	0.95	0.97	0.94	0.90	0.90	0.89	0.95	0.93	0.88	0.87	0.92$_{\pm .001}$
d = 10	0.98	0.99	0.97	0.94	0.96	0.95	0.98	0.97	0.93	0.94	**0.96**$_{\pm .001}$
$\mathcal{R}$-VAE											
d = 2	0.95	0.97	0.89	0.68	0.64	0.56	0.88	0.85	0.71	0.64	**0.78**$_{\pm .002}$
d = 5	0.95	0.98	0.94	0.91	0.94	0.88	0.95	0.93	0.90	0.89	**0.93**$_{\pm .0008}$
d = 10	0.98	0.98	0.96	0.95	0.96	0.95	0.97	0.97	0.93	0.94	**0.96**$_{\pm .001}$

pared against their Euclidean counterpart. The geodesic interpolation is influenced by the metric tensor, which makes sure that shortest paths will generally avoid areas of low density. This can easily be seen in top left Fig. 2.5, where the geodesic curve follows a path along a high density region. Contrast this to the Euclidean straight line between the two points traversing a lower density region. Reconstructed images along the curves can be seen in the middle and bottom rows. Even in less apparent cases (top right Fig. 2.5), reconstructions of latent codes along geodesic curves generally provide smoother transitions between the curve endpoints as can be seen by comparing the middle right and bottom right sections of the figure.

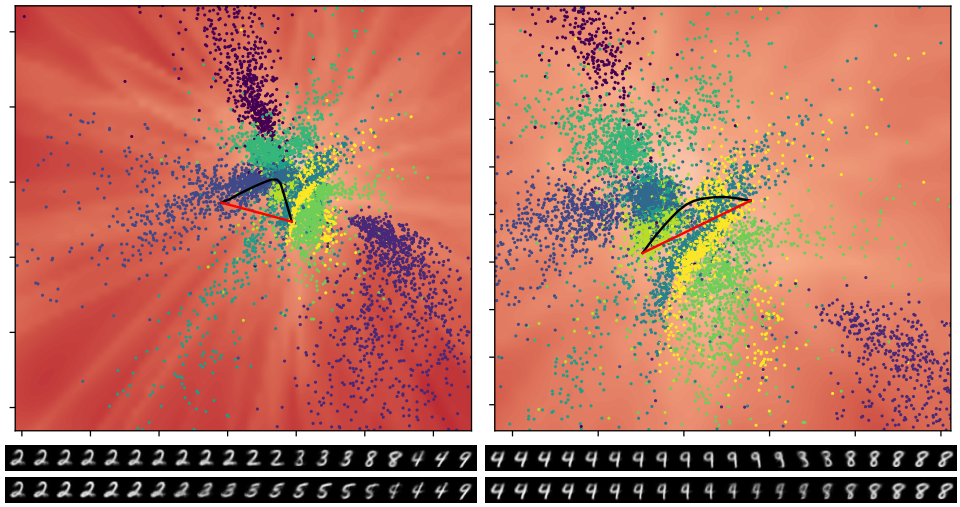


Figure 2.5. *Top*: Interpolations plotted in the latent space of $\mathcal{R}$-VAE. Black indicates a geodesic interpolant, red indicates a Euclidean interpolant. *Middle*: Images reconstructed along the geodesic interpolation. *Bottom*: Images reconstructed along the Euclidean interpolation. The latent codes are color-coded according to label. Best viewed in color.

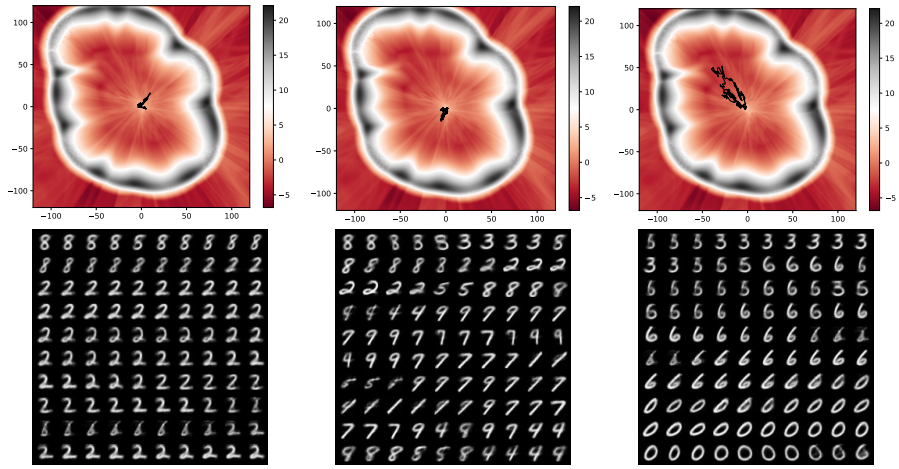
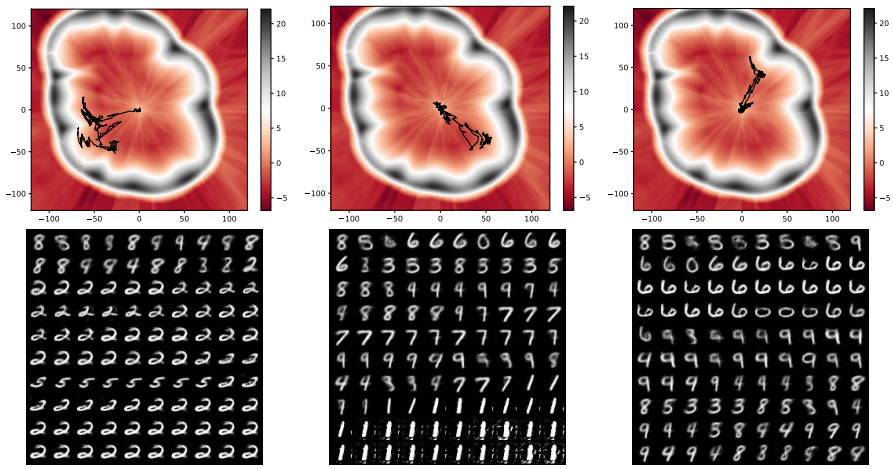


Figure 2.6. *Top*: Brownian motion runs on the learned latent manifold. *Bottom*: Corresponding sampled images. The sampler mostly stays in high density regions of the latent manifold. Best viewed in color.

Next, we investigate sampling from $\mathcal{R}$-VAE. In Sec. 2.4 we claimed that a Brownian motion prior coupled with the RBF regularization of the decoder variance network would yield samples that mostly avoid low density regions of the latent space. To



((a)) ((b)) ((c))

Figure 2.7. Brownian motion runs with artificially increased t (diffusion) parameter beyond the learned value. Note that the borders created by the metric tensor stop the sampler from exploring low density regions any further - the sampler either stops (a and b) or returns to regions of higher density (c). This effect is observed in the sampled images. Best viewed in color.

empirically prove this, we executed two sets of multiple sampling runs on the latent manifold. In the first set we ran Brownian motion with the learned prior parameters. These runs and the resulting images are displayed in Fig. 2.6. The random walks generally stay within high density regions of the manifold. Cases where they explore low density regions do exist but they are rare. The samples generally seem clear although sometimes their quality drops, especially when the sampler is transitioning between classes, where variance estimates are higher. This could potentially be rectified with a less aggressive deterministic warm-up scheme, which would result in more concentrated densities with thinner tails, although between-class variance estimates would likely still be higher compared to within-class ones. For the second set of the sampling runs, we increased the duration of the Brownian motion. These runs are displayed along with the sampled images in Fig. 2.7. The influence of the variance estimates on the metric tensor is clearly shown here. As the sampler is moving farther away from the latent code support, evaluations of the metric tensor increase making these regions harder to traverse. As a result the random walk either oscillates with decreased speed and stops close to the boundary (as in Figures 2.7(a) and 2.7(b)) or returns to higher density regions of the manifold. This clearly shows that $\mathcal{R}$-VAE mostly avoids the manifold mismatch problem.

2.6 Related work

Learned priors. In recent literature many works have identified the adverse effects of the KL divergence regularization when the prior is chosen to be a standard Gaussian. As such, there have been many approaches of learning a more flexible prior. Chen et al. [2016] propose learning an autoregressive prior by applying an Inverse Autoregressive transformation [Kingma et al., 2016] to a simple prior. Nalisnick and Smyth [2016] propose a non-parametric stick-breaking prior. Tomczak and Welling [2017] propose learning the prior as a mixture of variational posteriors. More recently, Bauer and Mnih [2018] present a rejection sampling approach with a learned acceptance function, while Klushyn et al. [2019] proposed a hierarchical prior through an alternative formulation of the objective.

Non-Euclidean latent space. Arvanitidis et al. [2018] was one of the first to analyze the latent space of a VAE from a non-Euclidean perspective. This work was inspired by Tosi et al. [2014] that studied the Riemannian geometry of the Gaussian process latent variable model [Lawrence, 2005b]. Arvanitidis et al. [2018] train a Euclidean VAE and fit a latent Riemannian LAND distribution [Arvanitidis et al., 2016a] and show that this view of the latent space leads to more accurate statistical estimates, as well as better sample quality.

Since then, a number of other works have appeared in literature that propose learning non-Euclidean latent manifolds. Xu and Durrett [2018] and Davidson et al. [2018a] learn a VAE with a von Mises-Fisher latent distribution, which samples codes on the unit hypersphere. Similarly, Mathieu et al. [2019] and Nagano et al. [2019] extend VAEs to hyperbolic spaces. Mathieu et al. [2019] assume a Poincaré ball model as a latent space and present 2 generalizations of the Euclidean Gaussian distribution - a wrapped Normal and the Riemannian Normal distributions, of which only the latter is a maximum entropy generalization. In practice, they perform similarly. Nagano et al. [2019] assume a Lorentz hyperbolic model as a latent space and also present a wrapped Normal generalization of the Gaussian. While these works have correctly identified the problem of the standard Gaussian not being a truly uninformative prior, due to the Euclidean assumption, they have proposed approaches which are designed for observations with known geometries. Most of the time, however, this information is not available and a more general framework for learning geometrically informed VAEs is needed. In response to this, Skopek et al. [2019] propose VAEs with the latent space modelled as a product of constant curvature manifolds, where each component curvature is learned. While more general than a model with a fixed curvature latent manifold, this framework still requires the specification of number of component manifolds along with the sign of their respective curvature. Finally, similar to our approach, Li et al. [2019] and Rey et al. [2019] both propose the heat kernel as a variational family representing a Brownian motion process on a Riemannian manifold.

They test their approaches on a priori chosen manifolds.

2.7 Conclusion

In this paper we presented VAEs with Riemannian manifolds as latent spaces and proposed a Riemannian generalization of the Gaussian along with an efficient sampling scheme. We show that the pull-back metric informs distances in the latent space, remaining invariant to reparameterizations. We further make explicit the relationship between uncertainty estimation and proper latent geometry and qualitatively show that geometrically informed priors avoid manifold mismatch by drawing samples from the image of the manifold in the latent space. Quantitatively, we show that our approach outperforms Euclidean VAEs both in an unsupervised learning task and a classification task, especially in low latent space dimensions.

CHAPTER 3

Pulling back information geometry

This chapter is adapted from Arvanitidis et al. [2021a]

Authors: Georgios Arvanitidis[1], Miguel González-Duque[1], Alison Pouplin[1], Dimitris Kalatzis[1], Søren Hauberg[1]

Abstract

Latent space geometry has shown itself to provide a rich and rigorous framework for interacting with the latent variables of deep generative models. The existing theory, however, relies on the decoder being a Gaussian distribution as its simple reparametrization allows us to interpret the generating process as a random projection of a deterministic manifold. Consequently, this approach breaks down when applied to decoders that are not as easily reparametrized. We here propose to use the Fisher-Rao metric associated with the space of decoder distributions as a reference metric, which we pull back to the latent space. We show that we can achieve meaningful latent geometries for a wide range of decoder distributions for which the previous theory was not applicable, opening the door to 'black box' latent geometries.

3.1 Introduction

Generative models such as *variational autoencoders (VAEs)* [Kingma and Welling, 2014, Rezende et al., 2014] and *generative adversarial networks (GANs)* [Goodfellow et al., 2014] provide state-of-the-art density estimators for high dimensional data. The underlying assumption is that data $\boldsymbol{x} \in \mathcal{X}$ lie near a low-dimensional manifold $\mathcal{M} \subset \mathcal{X}$, which is parametrized through a low-dimensional *latent representation* $\boldsymbol{z} \in \mathcal{Z}$. As data is finite and noisy, we only recover a probabilistic estimate of the true

[0]Equal contribution.

manifold, which, in VAEs, is represented through a decoder distribution $p(\boldsymbol{x}|\boldsymbol{z})$. Our target is the geometry of this *random manifold*.

The geometry of the manifold has been shown to carry great value when systematically interacting with the latent representations, as it provides a stringent solution to the *identifiability problem* that plagues latent variable models [Arvanitidis et al., 2018, Hauberg, 2018c, Tosi et al., 2014]. For example, this geometry has allowed VAEs to discover latent evolutionary signals in proteins [Detlefsen et al., 2020], provide efficient robot controls [Beik-Mohammadi et al., 2021, Chen et al., 2018b, Scannell et al., 2021], improve latent clustering abilities [Arvanitidis et al., 2018, Yang et al., 2018] and more. The fundamental issue with these geometric approaches is that the studied manifold is inherently a stochastic object, but classic differential geometry only supports the study of *deterministic* manifolds. To bridge the gap, Eklund and Hauberg [2019] have shown how VAEs with a Gaussian decoder family can be viewed as a random projection of a deterministic manifold, thereby making the classic theories applicable to the random manifold.

A key strength of VAEs is that they can model data from diverse modalities through the choice of decoder distribution $p(\boldsymbol{x}|\boldsymbol{z})$. For discrete data, we use categorical decoders, while for continuous data we may opt for a Gaussian, a Gamma or whichever distribution best suits the data. However, for non-Gaussian decoders, there exists no useful approach for treating the associated random manifold as deterministic, which prevents us from systematically interacting with the latent representations without being subjected to identifiability issues. This limitation motivates the current work.

In this paper, we provide a general framework that allows us to interact with the geometry of almost any random manifold. The key, and simple idea is to reinterpret the decoder as spanning a deterministic manifold in the space of probability distributions $\mathcal{H}$, rather than a random manifold in the observation space (see Fig. 3.1). Calling on classical *information geometry* [Amari, 2016, Nielsen, 2020], we show that the learned manifold is a Riemannian manifold of $\mathcal{H}$, and provide the corresponding computational tools. The approach is applicable to any family of decoders for which the KL-divergence can be differentiated, allowing us to work with a wide range of models from a single codebase.

3.2 The geometry of generative models

As a starting point, consider the deterministic generative model given by a prior $p(\boldsymbol{z})$ and a decoder $f : \mathcal{Z} = \mathbb{R}^d \to \mathcal{X} = \mathbb{R}^D$, which is assumed to be a smooth immersion. The latent representation $\boldsymbol{z}$ of an observation $\boldsymbol{x}$ is generally not *identifiable*, meaning that one can recover different latent representations that give rise to equally good density estimates. For example, let $g : \mathcal{Z} \to \mathcal{Z}$ be a smooth invertible function such that $\boldsymbol{z} \sim p(\boldsymbol{z}) \Leftrightarrow g(\boldsymbol{z}) \sim p(\boldsymbol{z})$, then the latent representation $g(\boldsymbol{z})$ coupled with the

decoder $f \circ g^{-1}$ gives the same density estimate as $\boldsymbol{z}$ coupled with f [Hauberg, 2018c]. Practically speaking, the identifiability issue implies that it is improper to view the latent space $\mathcal{Z}$ as being Euclidean, as any reasonable view of $\mathcal{Z}$ should be invariant to reparametrizations g.

The classic geometric solution to the identifiability problem is to define any quantity of interest in the observation space $\mathcal{X}$ rather than the latent space $\mathcal{Z}$. For example, the length of a curve $\gamma : [0, 1] \to \mathcal{Z}$ in the latent space can be defined as its length measured in $\mathcal{X}$ on the manifold $\mathcal{M} = f(\mathcal{Z})$ with $N \to +\infty$ as:

$$\begin{aligned}\mathcal{L}(\gamma) &= \sum_{n=1}^{N-1} \|f(\gamma(t_{n+1})) - f(\gamma(t_n))\| = \int_0^1 \|\dot{f}(\gamma(t))\| dt \\ &= \int_0^1 \sqrt{\dot{\gamma}(t)^\top J_f(\gamma(t))^\top J_f(\gamma(t)) \dot{\gamma}(t)} dt, \end{aligned} \tag{3.1}$$

where $t_n = {}^n/_N$ and $t_{n+1} = {}^{n+1}/_N$ and we used the chain rule $\partial_t f(\gamma(t)) = J_f(\gamma(t))\dot{\gamma}(t)$ with $\dot{\gamma}(t) = \partial_t \gamma(t)$ being the curve derivative, and $J_f(\gamma(t)) \in \mathbb{R}^{D \times d}$ the Jacobian of f at $\gamma(t)$. This construction shows how we may calculate lengths in the latent space with respect to the metric of the observation space, which is typically assumed to be the Euclidean, but other options exist [Arvanitidis et al., 2021b]. In this way, the symmetric positive definite matrix $J_f(\gamma(t))^\top J_f(\gamma(t))$ is denoted by $G(\gamma(t)) \in \mathbb{R}^{d \times d}_{\succ 0}$ and captures the geometry of $\mathcal{M}$ in $\mathcal{Z}$. This is known as the *pullback metric* as it pulls the Euclidean metric from $\mathcal{X}$ into $\mathcal{Z}$. As the Jacobian spans the d-dimensional tangent space at the point $\boldsymbol{x} = f(\boldsymbol{z})$, we may interpret $G(\boldsymbol{z})$ as an inner product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle_G = \boldsymbol{u}^\top G(\boldsymbol{z}) \boldsymbol{v}$ over this tangent space, given us all the ingredients to define *Riemannian manifolds*:

Definition 6. *A Riemannian manifold is a smooth manifold $\mathcal{M}$ together with a Riemannian metric $G(\boldsymbol{z})$, which is a positive definite matrix that changes smoothly throughout space and defines an inner product on the tangent space $\mathcal{T}_{\boldsymbol{z}}\mathcal{M}$.*

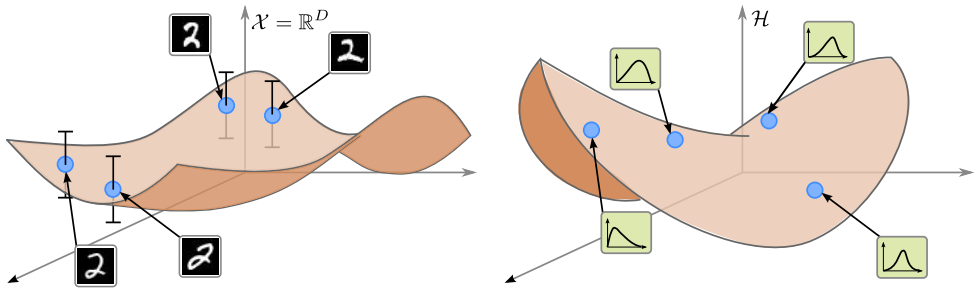


Figure 3.1. Traditionally (left), we view the learned manifold as a stochastic manifold in the observation space. We propose (right) to view the learned manifold as a deterministic manifold embedded in the space of decoder distributions, which is equipped with a Fisher-Rao metric based on information geometry.

We see that the decoder naturally spans a Riemannian manifold and the latent space $\mathcal{Z}$ can be considered as the *intrinsic coordinates*. Technically, we can consider any Euclidean space as the intrinsic coordinates of an abstract $\mathcal{M}$ using a suitable metric $G(\boldsymbol{z})$, which is implicitly induced by an abstract f. Since the Riemannian length of a latent curve equation 3.1, by construction, is invariant to reparametrizations, it is natural to extend this view with a notion of *distance*. We say that the distance between two points $\boldsymbol{z}_0, \boldsymbol{z}_1 \in \mathcal{Z}$ is simply the length of the shortest connecting path, $\text{dist}(\boldsymbol{z}_0, \boldsymbol{z}_1) = \min_\gamma \text{L}(\gamma)$. Calculating distances implies finding the shortest path. One can show [Gallot et al., 2004] that length minimizing curves also have minimal *energy*:

$$\mathcal{E}(\gamma) = \int_0^1 \|\dot{f}(\gamma(t))\|^2 dt = \int_0^1 \dot{\gamma}(t)^\top G(\gamma(t))\dot{\gamma}(t) dt, \tag{3.2}$$

which is a locally convex functional. Shortest paths can then be found by direct energy minimization [Yang et al., 2018] or by solving the associated system of ordinary differential equations (ODEs) [Arvanitidis et al., 2019, Hennig and Hauberg, 2014] (see supplementary materials for additional details).

3.2.1 Stochastic decoders

As previously discussed, deterministic decoders directly induce a Riemannian geometry in the latent space. However, most models of interest are stochastic and there is significant evidence that this stochasticity is important to faithfully capture the intrinsic structure of data [Hauberg, 2018c]. When the decoder is a smooth stochastic process, e.g. as in the Gaussian Process Latent Variable Model (GP-LVM) [Lawrence, 2005a], Tosi et al. [2014] laid the foundations for modeling a stochastic geometry. Most contemporary models, such as VAEs, assume independent noise, making this theory inapplicable. Arvanitidis et al. [2018] proposed an extension of this stochastic geometry to VAEs with Gaussian decoders, which take the form

$$\begin{aligned} f(\boldsymbol{z}) &= \mu(\boldsymbol{z}) + \sigma(\boldsymbol{z}) \odot \boldsymbol{\epsilon} \\ &= \begin{bmatrix} \mathbb{I}_D & \text{diag}(\boldsymbol{\epsilon}) \end{bmatrix} \begin{bmatrix} \mu(\boldsymbol{z}) \\ \sigma(\boldsymbol{z}) \end{bmatrix} = P_\varepsilon \; h(\boldsymbol{z}), \end{aligned} \tag{3.3}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_D)$. Here we have written the Gaussian decoder in its reparametrized form. This can be viewed as a random projection of a deterministic manifold spanned by h with projection matrix P_ϵ [Eklund and Hauberg, 2019], which can easily be given a geometry. The associated Riemannian metric,

$$G(\boldsymbol{z}) = J_\mu(\boldsymbol{z})^\top J_\mu(\boldsymbol{z}) + J_\sigma(\boldsymbol{z})^\top J_\sigma(\boldsymbol{z}), \tag{3.4}$$

gives shortest paths that follow the data as distances grow with the model uncertainty [Arvanitidis et al., 2018, Hauberg, 2018c]. An example of a shortest path $\gamma(t) \in \mathcal{Z}$

computed under this metric is shown in Fig. 3.2 and the respective curve on the corresponding expected manifold $\mu(\gamma(t)) \in \mathcal{M} \subset \mathcal{X}$.

Previous work has, thus, focused on *pulling back* the Euclidean metric from the observation space to the latent space using the reparameterization of the Gaussian decoder. This is, however, intrinsically linked with the simple reparameterization of the Gaussian, and this strategy can only extend to location-scale distributions. We propose an alternative, principled way of dealing with stochasticity by changing the focus from the observation space $\mathcal{X}$ to the parameter space $\mathcal{H}$ associated to the distribution of the decoder, leveraging the metrics defined in classical information geometry.

3.3 Information geometric latent metric

So far we have seen how we can endow the latent space $\mathcal{Z}$ with meaningful distances only when our stochastic decoders are reparameterizable and their codomain is the observation space $\mathcal{X}$. Ideally, we would like a more general framework of computing shortest path distances for a more general class of distributions.

We first note that the codomain of a VAE decoder is the parameter space $\mathcal{H}$ of a probability density function. In particular, depending on the type of data we specify a likelihood $p(\boldsymbol{x}|\eta)$ with parameters $\eta \in \mathcal{H}$, which we can rewrite as $p(\boldsymbol{x}|\boldsymbol{z})$ using the mapping $h : \mathcal{Z} \to \mathcal{H}$.

With this in mind, we can ask what is a natural distance in the latent space $\mathcal{Z}$ between two infinitesimally near points $\boldsymbol{z}_1$ and $\boldsymbol{z}_2 = \boldsymbol{z}_1 + \epsilon$ when measured in $\mathcal{H}$. Since our latent codes map to distributions we can define the (infinitesimal) distance through the KL-divergence:

$$\text{dist}^2(\boldsymbol{z}_1, \boldsymbol{z}_2) = D_{\text{KL}}(p(\boldsymbol{x}|\boldsymbol{z}_1)||p(\boldsymbol{x}|\boldsymbol{z}_2)). \tag{3.5}$$

So we can define the length of a curve $\gamma : [0, 1] \to \mathcal{Z}$ as

$$\mathcal{L}(\gamma) = \lim_{N\to\infty} \sum_{n=1}^{N-1} D_{\text{KL}}(p(\boldsymbol{x}|\gamma(t_n)||p(\boldsymbol{x}|\gamma(t_{n+1})))^{\frac{1}{2}}, \tag{3.6}$$

and distances could be defined as before. This would satisfy our desiderata of a deterministic notion of similarity in the latent space that is applicable to wide range of decoder distributions.

This construction may seem arbitrary, but in reality it carries deeper geometric meaning. *Information geometry* [Nielsen, 2020] considers families of probabilistic densities $p(\boldsymbol{x}|\eta)$ as represented by their parameters $\eta \in \mathcal{H}$, such that $\mathcal{H}$ is constructed as a statistical manifold equipped with the Fisher-Rao metric, which infinitesimally coincides

with the KL divergence in equation 3.5. This is known to be a Riemannian metric over $\mathcal{H}$ that takes the following form:

$$I_{\mathcal{H}}(\eta) = \int_{\mathcal{X}} [\nabla_\eta \log p(\boldsymbol{x}|\eta) \nabla_\eta \log p(\boldsymbol{x}|\eta)^\top] p(\boldsymbol{x}|\eta) d\boldsymbol{x}. \tag{3.7}$$

When the parameter space $\mathcal{H}$ is equipped with this metric, we call it a *statistical manifold*.

Definition 7. *A statistical manifold consists of the parameter space $\mathcal{H}$ of a probability density function $p(\boldsymbol{x}|\eta)$ equipped with the Fisher-Rao information matrix $I_{\mathcal{H}}(\eta)$ as a Riemannian metric.*

Note that the geometry induced by the Fisher-Rao metric is predefined and can be seen as a modeling decision, since it is related to the chosen likelihood and does not change with data.

As previously mentioned, a known result in Information Geometry is that the Fisher-Rao metric coincides with the KL-divergence locally [Amari, 2016, Nielsen, 2020]:

Proposition 1. *The Fisher-Rao metric is the second order approximation of the KL-divergence between perturbed distributions:*

$$D_{\mathrm{KL}}(p(\boldsymbol{x}|\eta)||p(\boldsymbol{x}|\eta + \delta\eta)) = \frac{1}{2} \delta\eta^\top I_{\mathcal{H}}(\eta) \delta\eta + o(\delta\eta^2). \tag{3.8}$$

The central idea put forward in this paper is to consider the decoder as a map $h : \mathcal{Z} \to \mathcal{H}$ instead of $f : \mathcal{Z} \to \mathcal{X}$, and let $\mathcal{H}$ be equipped with the appropriate Fisher-Rao metric. The VAE can then be interpreted as spanning a manifold $h(\mathcal{Z})$ in $\mathcal{H}$ and the latent space $\mathcal{Z}$ can be endowed with the corresponding metric. We detail this approach in the following section.

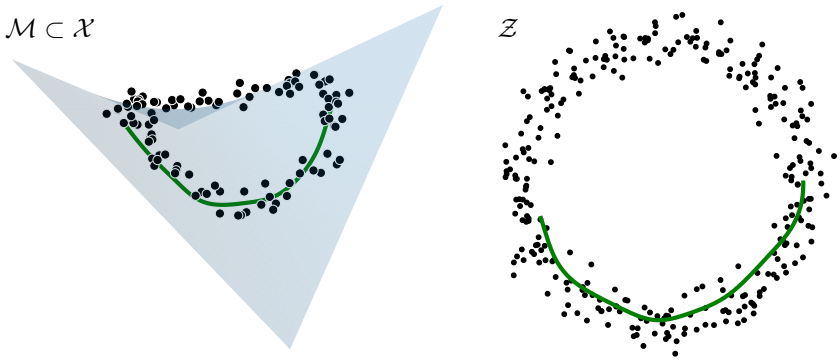


Figure 3.2. A conceptual example of a Riemannian manifold $\mathcal{M} = \mu(\mathcal{Z})$ lying in $\mathcal{X}$ and the corresponding latent space $\mathcal{Z}$, together with an associated shortest path.

3.3.1 The Riemannian pull-back metric

Our construction implies that the length of a latent curve $\gamma : [0, 1] \to \mathcal{Z}$ when mapped through h can be measured in the parameter space $\mathcal{H}$ using the Fisher-Rao metric therein as

$$\mathcal{L}(\gamma) = \int_0^1 \sqrt{\partial_t h(\gamma(t))^\top I_{\mathcal{H}}(h(\gamma(t)))\partial_t h(\gamma(t))} dt, \tag{3.9}$$

with M the pullback metric:

Proposition 2. *Let $h : \mathcal{Z} \to \mathcal{H}$ be an immersion that parameterizes the likelihood. Then, the latent space $\mathcal{Z}$ is equipped with the Riemannian pull-back metric $G(\boldsymbol{z}) = J_h^\top(\boldsymbol{z}) I_{\mathcal{H}}(h(\boldsymbol{z})) J_h(\boldsymbol{z})$.*

Proof. See appendix, Prop. 7. □

Note that instead of considering the parameters $\eta \in \mathcal{H}$ of the probabilistic density function $p(\boldsymbol{x}|\eta)$ that approximates the data, we can consider the latent variable $\boldsymbol{z}$ as the actual parameters of the model. This view is equivalent to the one explained above, and the corresponding pull-back metric is directly the Fisher-Rao metric endowed in the latent space $\mathcal{Z}$:

Proposition 3. *The pullback metric $G(\boldsymbol{z})$ is identical to the Fisher-Rao metric obtained over the parameter space $\mathcal{Z}$ as $G(\boldsymbol{z}) = \int_{\mathcal{X}} \left[\nabla_{\boldsymbol{z}} \log p(\boldsymbol{x}|\boldsymbol{z}) \nabla_{\boldsymbol{z}} \log p(\boldsymbol{x}|\boldsymbol{z})^\top\right] p(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{x}$.*

Proof. See appendix, Prop. 8. □

Therefore, pulling back the Fisher-Rao metric from $\mathcal{H}$ into $\mathcal{Z}$ enables us to compute length minimizing curves which are indentifiable (see Sec. 3.2). The advantange of this approach is that it applies to any type of decoders and data, as the actual distance is measured over the manifold spanned by h in the parameter space $\mathcal{H}$. So shortest paths between probability distributions move optimally on this manifold while taking the geometry of $\mathcal{H}$ into account through the Fisher-Rao metric.

Computing shortest paths directly in $\mathcal{H}$ need not result in a sensible sequence of probability density functions $p(\boldsymbol{x}|\eta)$. To ensure that the shortest paths computed under our metric stay within the support of the data, we carefully design our decoder h to extrapolate to uncertain distributions outside the support of the data (see supplements for additional details).

In Fig. 3.3 we compare a shortest path $\gamma : [0, 1] \to \mathcal{Z}$ under the proposed metric $G(\boldsymbol{z})$ against a curve $c : [0, 1] \to \mathcal{H}$ with minimal length. We consider a Gaussian likelihood with isotropic covariance. We show the resulting sequence of means for

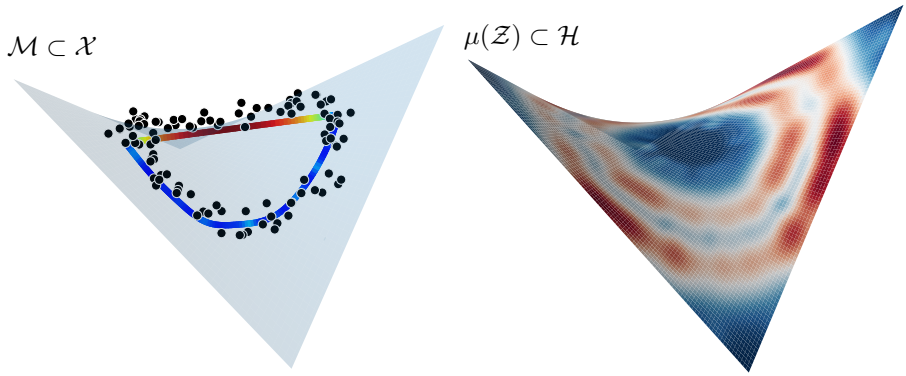


Figure 3.3. *Left*: The optimal $\gamma(t)$ (blue curve) under $G(\boldsymbol{z})$ results to distributions that respect the structure of data, while the curve $c(t)$ (red-green curve) with minimal length in $\mathcal{H}$ does not as it leaves $\mathcal{M}$. Red and green signal high and low variance respectively. *Right*: A part of the spanned manifold $h(\mathcal{Z}) = [\mu(\mathcal{Z}), \sigma(\mathcal{Z})] \in \mathcal{H}$ colored by $|G(\boldsymbol{z})|$. Note that we design $\sigma(\boldsymbol{z})$ to increase far from the data, which ensures that $\gamma(t)$ stays within their support.

both interpolants color-coded by the corresponding variances. As expected $c(t)$ does not take into account the given data, but only respects the geometry of $\mathcal{H}$ implied by the likelihood.

3.3.2 Efficient shortest path computation

An essential task in computational geometry is to compute shortest paths. This can be achieved by minimizing curve energy equation 3.2 or solving the corresponding system of ODEs (see supplementary material). The latter, however, requires inordinate computational resources, since the evaluation of the system relies on the Jacobian of the decoder and its derivatives.

Bearing in mind that the metric is an approximation of the KL divergence between perturbations equation 3.8, the energy is directly expressed as a sum of KL divergence terms along a discretized curve γ:

$$\mathcal{E}(\gamma) \propto \lim_{N \to \infty} \sum_{n=1}^{N-1} D_{\mathrm{KL}}(p(\boldsymbol{x}|\gamma(t_n)) || p(\boldsymbol{x}|\gamma(t_{n+1}))). \tag{3.10}$$

The proof can be found in the appendix, Prop. 6. A simple algorithm for computing shortest paths is to minimize equation 3.10 with respect to the parameters of the curve γ. Here we represent γ as a cubic spline with fixed end-points. Then standard free-form optimization can be applied to minimize this energy.

3.3.3 Example: categorical decoders

The motivation for our approach is that, while several options for decoders exist in VAEs depending on the type of the given data, we could only capture and use the learned geometry in a principled way with Gaussian decoders. Our proposed methodology is more general.

For a constructive example, assume that $\boldsymbol{x}$ is a categorical variable. We can select a generalized Bernoulli likelihood $p(\boldsymbol{x}|\boldsymbol{z})$, such that $h(\boldsymbol{z}) = (\eta_1, \cdots, \eta_D)$ where each η_i represents the probability of x_i being 1. Thus, the parameters η lie on the unit simplex $\mathcal{H}$, and the distance under the corresponding Fisher-Rao metric between points on the simplex coincides with the spherical distance between the points $\sqrt{\eta}$ on the unit sphere,

$$\operatorname{dist}(\eta, \eta') = \arccos\left(\sqrt{\eta}^\top \sqrt{\eta'}\right). \tag{3.11}$$

We derive in detail this previously known result in the supplementary materials.

Given a curve $\gamma : [0, 1] \to \mathcal{Z}$ we can approximate the energy by using the small angle approximation $\cos\theta \approx 1 - \theta^2/2 \Leftrightarrow \theta^2 \approx 2 - 2\cos\theta$ to give

$$\mathcal{E}(\gamma) = \sum_{n=1}^{N-1} \left(2 - 2\sqrt{h(\gamma(t_n))}^\top \sqrt{h(\gamma(t_{n+1}))}\right), \tag{3.12}$$

for sufficiently fine discretization with $t_n = n/N$ and $t_{n+1} = n+1/N$. This gives a particular simple expression for the energy, which we can minimize in order to compute the shortest path.

3.3.4 Black-box random geometry

In general, we can derive suitable expressions for computing metrics and energies for families of decoders, doing so is tedious, error-prone and time-consuming. This limits the practical use of the developed theory.

Drawing inspiration from *black-box variational inference* [Ranganath et al., 2014], we propose a notion of *black-box random geometry*. Assume that we have access to a differentiable KL divergence for our choice of decoder distribution. We can then apply the methodology presented in Sec. 3.3.2 to compute shortest paths.

In practice, modern libraries such as PyTorch [Paszke et al., 2019] have this functionality implemented for several distributions. When we do not have closed-form expression for the KL divergence, we can resort to Monte Carlo estimates thereof. More specifically, we can estimate the KL divergence by generating samples from the likelihood based on the re-parametrization trick, which allows us to get derivatives with automatic differentiation.

Interestingly, apart from finding the shortest path through the KL formulation, we can also approximate the actual metric tensor $G(\boldsymbol{z})$. As we have discussed above, evaluating explicitly this metric is not a trivial task in many cases. One problem is that we need access to the Jacobian of the parametrization h, which is typically a deep neural network, so the computation is not always straightforward. Alternatively, one could use that the Fisher-Rao metric is the Hessian of the KL-divergence equation 3.8, but such approaches fare poorly with current tools for automatic differentiation, where higher-order derivatives are often incompatible with batching. Furthermore, the Fisher-Rao metric itself may be intractable depending on the chosen likelihood $p(\boldsymbol{x}|\eta)$. Nevertheless, we show that the KL formulation equation 3.8 allows us to approximate the latent metric as:

Proposition 4. *We define perturbations vectors as* $\delta e_i = \varepsilon \cdot \mathbf{e_i}$*, with* $\varepsilon \in \mathbb{R}_+$ *a small infinitesimal quantity, and* $\mathbf{e_i}$ *a canonical basis vector in* $\mathbb{R}^d$*. For clarity, we rename* $D_{\mathrm{KL}}(p(\boldsymbol{x}|\boldsymbol{z})||p(\boldsymbol{x}|\boldsymbol{z}+\delta\boldsymbol{z})) = KL_{\boldsymbol{z}}(\delta\boldsymbol{z})$ *and we note* $G_{ij} = G_{ji}$ *the components of* $G(\boldsymbol{z})$*. We can then approximate by a system of equations the diagonal and non-diagonal elements of the metric:*

$$
\begin{aligned}
G_{ii} &\approx 2\ KL_{\boldsymbol{z}}(\delta\mathbf{e_i})/\varepsilon^2 \\
G_{ji} &\approx \left(KL_{\boldsymbol{z}}(\delta\mathbf{e_i}+\delta\mathbf{e_j}) - KL_{\boldsymbol{z}}(\delta\mathbf{e_i}) - KL_{\boldsymbol{z}}(\delta\mathbf{e_j})\right)/\varepsilon^2.
\end{aligned}
$$

See Prop. 10 in the appendix for a proof. Note that this formulation only requires h to be a smooth immersion. This is particularly useful, as the metric is used for other purposes on a Riemannian manifold and not exclusively for computing shortest paths. For example, relying on $G(\boldsymbol{z})$ we can compute the exponential map by solving the corresponding ODE system as an initial value problem. Assuming a fully differentiable KL divergence, then the approximated metric is also differentiable. This is all that is required for practical usage of differential geometry, and thus, we have a reasonable notion of *black-box random geometry*.

3.4 Experiments

3.4.1 Pulling back Euclidean and Fisher-Rao metric with Gaussian decoders

We start our experiments by comparing our proposed way of inducing geometry in latent spaces with the existing theory: pulling back the Euclidean metric using a stochastic Gaussian decoder (see equation 3.4). We also include in this comparison the effect of regularizing the uncertainty quantification in the learned geometries. In this regularization, we use transition networks [Detlefsen et al., 2019b] to ensure high

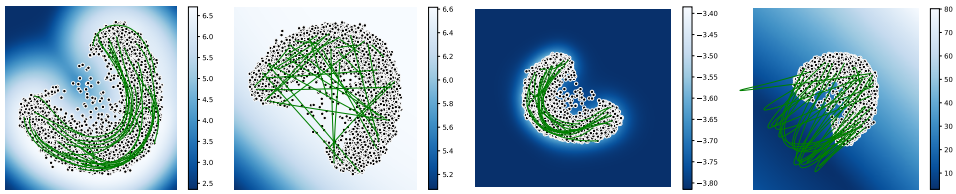


Figure 3.4. Pulling back the Euclidean and Fisher-Rao metrics with Gaussian decoders. Left to right: Euclidean pull-back with regularized uncertainty, Euclidean pull-back with a NN to model uncertainty, Fisher-Rao pull-back with regularized uncertainty, Fisher-Rao pull-back with a NN to model uncertainty.

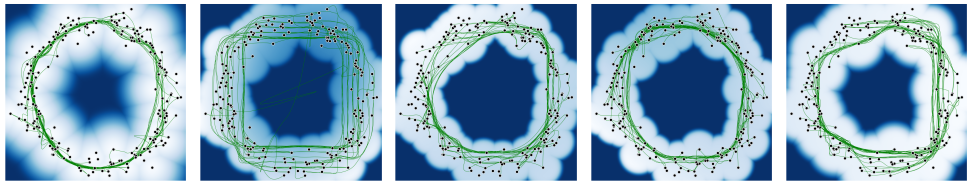


Figure 3.5. Pulling back the metric from different parameter spaces. From left to right: Normal, Bernoulli, Beta, Dirichlet and Exponential. White areas represent low entropy of the decoded distribution, while blue areas represent higher entropy. Notice that the Bernoulli latent space is darker blue (i.e. more entropic) because distributions with parameters around $1/2$ are near uniform.

uncertainty outside the support of the data (see Sec. B.3.2 in the supplementary material).

In this experiment, we train four VAEs on a subset of the MNIST dataset composed of only the digits with label 1. Two of these VAEs implement a standard Gaussian decoder, and we induce a metric in the latent space by pulling the Euclidean metric back using the Jacobian of the decoder. In the other two, we consider the output of the decoder as lying in a statistical manifold and approximate the pullback of the Fisher-Rao metric by using the KL divergence locally. In each of these two sets, one of the decoders implements the uncertainty regularization described above.

Fig. 3.4 shows the latent spaces of these four decoders, illuminated by the volume measure. In each of this latent spaces, we analyze the geometry induced by the respective pullbacks by computing and plotting several shortest paths. This figure illustrates two key findings: (1) Our approach is on par with the existing literature in learning geometric structure, which can be seen by comparing the first and third latent spaces (Euclidean vs. Fisher Rao, respectively), and (2) Performing uncertainty regularization plays an instrumental role on learning a sensible geometric structure, which can be seen when comparing the first and second latent spaces (both coming from the Euclidean pullback, with and without regularization respectively), and similarly for the third and fourth.

3.4.2 The Fisher-Rao pullback metric for various distributions with toy data

For our second experiment, we induced a geometry on a known latent space (given by noisy circular data in $\mathcal{Z}_{\text{toy}} = \mathbb{R}^2$) by *pulling back* the Fisher-Rao metric from the parameter space of different distributions, showcasing the potential for computing shortest paths efficiently, even in non-Gaussian settings. The statistical manifolds from which we pull the metric are associated with multivariate versions of the Normal, Bernoulli, Beta, Dirichlet and Exponential distributions. For this approximation to follow the support of the data we need to ensure that our mapping $\mathcal{Z}_{\text{toy}} \to \mathcal{H}$ extrapolates to high uncertainty outside our training codes (see Fig. 3.4). To do so, we perform uncertainty regularization for each one of the decoded distributions (see supplementary materials for implementation details).

In Fig. 3.5 we show the toy latent space alongside several shortest paths computed using the pullback of the Fisher-Rao metric from the statistical manifolds associated with the Gaussian, Bernoulli, Beta, Dirichlet and Exponential distributions. We parametrize the curves as cubic splines and minimize their energy using automatic differentiation (see Sec. 3.3.2). These results show that the approximated pulled-back metric induces a meaningful geometry in this latent space, which recovers the true circular structure of the data. In the case of the Bernoulli distribution, we notice that some of the paths fail to converge. We hypothesize that our uncertainty regularization (which decodes to the uniform distribution outside the support) is not strong enough since Bernoulli distributions with parameters close to 1/2 are already highly entropic.

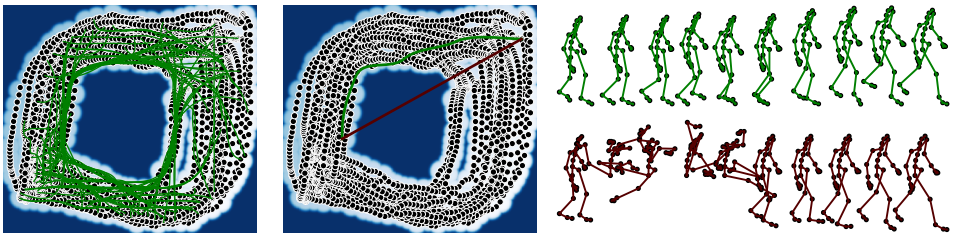


Figure 3.6. *Left:* Geodesics in the latent space of a von Mises-Fisher decoder. *Middle*: Shortest path (green) vs. linear (red). *Right*: decoding the shortest path (green) vs. the linear interpolation (red) as poses (i.e. the product of von Mises-Fisher distributions). Our path follows the trend of the data manifold, while the linear path traverses regions with no data support.

3.4.3 Motion capture data with products of von Mises-Fisher distributions

As a further demonstration of our black-box random geometry, we consider a model of human motion capture data. Here we observe a time series, where each time point represent a 'skeleton' corresponding to a human pose. As only pose, and not shape, changes over time, individual limbs on the body only change position and orientation, but not length. Each limb is then a point on a sphere in $\mathbb{R}^3$ with radius given by the limb length. Following Tournier et al. [2009] we view the skeleton representation space as a product of spheres. From this, we build a VAE where the decoder distribution is a product of von Mises-Fisher distributions. To ensure a sensible uncertainty estimates in the decoder, we enforce that the concentration parameter extrapolate to a small constant.

In this case, we do not have easily accessible Fisher-Rao metrics, so we lean on the KL formulation from Sec. 3.3.4. Since, the KL does not have a closed-form expression for the von Mises-Fisher distribution, we resort to a Monte Carlo estimate thereof. This is realisable with off-the-shelf tools [Davidson et al., 2018b].

Fig. 3.6 shows the latent representation of a motion capture sequence of a person walking (Seq. `69_06` from `http://mocap.cs.cmu.edu/`) with shortest paths superimposed. We see that our paths follow the trend of the data, and reflect the underlying periodic nature of the observed walking motion. We pick two random points in the latent space, and traverse both the shortest path and the straight line implied by a Euclidean interpretation of the latent space. As we traverse, we sample from the decoder distribution, thereby producing two new motion sequences, which appear in Fig. 3.6. As can be seen, the straight line traverses uncharted territory of the latent space and end up creating an implausible motion. This is in contrast to the shortest path, that consistently generates meaningful poses.

3.4.4 Numerical approximation of the Fisher-Rao pullback metric

Prop. 4 provide an approximation to the metric and we test its accuracy as per equation 3.8. We discretize the latent space for the just-described von Mises-Fisher decoder and, for each $\boldsymbol{z}$ in this grid, we both approximate $G(\boldsymbol{z})$ and compute the expected value of $\|\mathrm{KL}(p(\boldsymbol{x}|\boldsymbol{z}), p(\boldsymbol{x}|\boldsymbol{z}+\delta\boldsymbol{z})) - \frac{1}{2}\delta\boldsymbol{z}^\top G(\boldsymbol{z})\delta\boldsymbol{z}\|$ for several samples of $\delta\boldsymbol{z}$, uniformly distributed around the circle of radius $\varepsilon = 0.1$. Notice that we do not have a ground truth to compare against, and that this error will always be off by $O(\delta\boldsymbol{z}^2)$. Fig. 3.7 shows the average error, where we can see that the approximate metric is well-estimated both within and outside the support of the data. The error, however, grows at the boundaries of the support, where the distribution is changing from a concentrated von Mises-Fisher to a uniform distribution. It is worth mentioning

that we observe some approximated metrics have negative determinant, showing that our numerical approximations are imprecise at the boundary. These results warrant further research on more stable ways of approximating pulled back metrics under our proposed approach.

3.4.5 Statistical models on manifolds

We demonstrate the usefulness of the approximated metrics, by fitting a distribution to data in the latent space, which requires normalization according to the measure induced by the metric. In particular, we fit a locally adaptive normal distribution (LAND) [Arvanitidis et al., 2016b], which extends the Gaussian distribution to learned manifolds. The probability density function is $\rho(\boldsymbol{z}) = C(\mu, \boldsymbol{\Sigma}) \cdot \exp\left(-0.5 \cdot \mathrm{Log}_{\mu}(\boldsymbol{z})^{\top} \Gamma \mathrm{Log}_{\mu}(\boldsymbol{z})\right)$, where $\mu \in \mathbb{R}^d$ is the mean, $\Gamma \in \mathbb{R}^{d \times d}_{\succ 0}$ is the precision matrix and $C(\mu, \Gamma)$ the normalization constant. The operator $\mathrm{Log}_{\mu}(\boldsymbol{z})$ returns the scaled initial velocity $\mathbf{v} = \dot{\gamma}(0) \in \mathbb{R}^d$ of the shortest connecting path with $\gamma(1) = \boldsymbol{z}$ and $||\mathbf{v}|| = \mathrm{Length}(\gamma)$. In Fig. 3.7 we show the LAND density on the learned latent representations under the approximated Riemannian metric from Sec. 3.4.4. Since shortest paths follow the data, so does the density ρ. See supplementary material for details.

3.4.6 Movie preferences via latent interpolants

In addition, we explored the latent space of the movie-users rating dataset MovieLens 25M (https://grouplens.org/datasets/movielens/25m/). In particular, we consider a Bernoulli VAE to model if a user has watched a movie among the 60 most popular in the dataset. Also, we considered only users who have seen less than 30

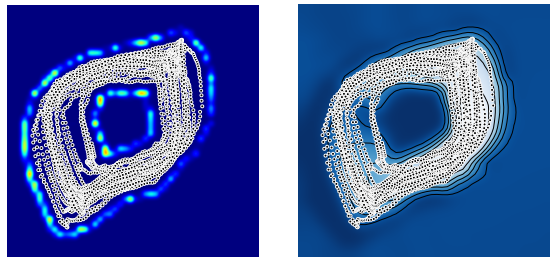


Figure 3.7. *Left*: Average error of the approximated metric in the von Mises-Fisher latent space. Darker colors indicate lower error (less than ε^2), while higher values are clear. *Right*: The LAND density well-adapts to the nonlinear structure of the latent representations due to the shortest paths behavior.

movies. The implementation and preprocessing details can be found in the supplementary material. Our VAE decodes to 60 Bernoulli parameters that are conditionally independent given the latent code $\boldsymbol{z}$, which state the likelihood that a given user has seen these movies. Latent codes in this space, then, can be seen as individual users with certain movie preferences.

We then computed the shortest path between two points by considering the pulled-back Fisher-Rao (see Sec. 3.3.2), and we compare against a straight line interpolation. We consider the cosine similarity of the decoded outputs. This cosine similarity measures whether two users (encoded as points in the latent space) have similar preferences according to our model. In Fig. 3.8 we see that our path follows users with similar movie preferences locally, while the linear interpolation failed to capture a local notion of preference.

We then computed the shortest path between two points by considering the pulled-back Fisher-Rao (see Sec. 3.3.2), and we compare against a straight line interpolation. We consider the cosine similarity of the decoded outputs. This cosine similarity measures whether two users (encoded as points in the latent space) have similar preferences according to our model. In Fig. 3.8 we see that our path follows users with similar movie preferences locally, while the linear interpolation failed to capture a local notion of preference.

3.5 Related work

The literature is rich on deterministic generative models such as autoencoders [Rumelhart et al., 1986b] and generative adversarial networks [Goodfellow et al., 2014],and a series of papers have investigated such deterministic decoders [Chen et al., 2018a, Laine, 2018, Shao et al., 2018]. However, our work is not applicable to this setting. As demonstrated in Sec. 3.4.1 stochasticity is essential to shape the latent space according to the data manifold. Hauberg [2018c] argues model uncertainty plays a role much akin to topology in classic geometry, in that it, practically, allows us to deviate from the Euclidean topology of the latent space.

Our constructions rely on information geometry and in particular Fisher-Rao metrics [Nielsen, 2020]. While our work is within the spirit of information geometry, it does not represent typical usage of this theory. Information geometry has been widely used in the context of optimisation with *natural gradients* [Martens, 2014, Martens and Grosse, 2015], Markov Chain Monte Carlo methods [Girolami and Calderhead, 2011] and hypothesis testing [Nielsen, 2020]. The key difference between natural gradients and our work is the space we wish to explore: in the case of the natural gradients, the shortest path is obtained on the space of the weights of the neural networks, while we aim to explore the latent space of a VAE. It can also be noted that Information geometry provides a rich family of alternative divergences over the

here-applied KL-divergence. We did not investigate their usage in our context.

To make use of the here-developed tools, we may lean on techniques for statistics on manifolds. These provide generalizations of a long list of classic statistical algorithms [Fletcher, 2011, Hauberg, 2016, Zhang and Fletcher, 2013]. We refer the reader to Pennec [2006b] for a gentle introduction to this line of research.

3.6 Conclusion and discussion

We have proposed a new approach for getting a well-defined and useful geometry in the latent space of generative models with stochastic decoders. The theory is easy to apply and readily generalize to a large family of decoder distributions. The latent geometry gives access to a series of operations on latent variables that are invariant to reparametrizations of the latent space, and therefore are not subject to a large class of identifiability issues. Such operational representations have already shown great value in applications ranging from biology [Detlefsen et al., 2020] to robotics [Scannell et al., 2021]. We have here focused on the Fisher-Rao metric, but other geometries over distributions may apply equally well, e.g. the Wasserstein geometry may be interesting to explore.

Limitations. The largest practical hurdle with the proposed methodology, is that it only works well for decoders with well-calibrated uncertainties. That is, the decoder should yield high entropy in regions of little training data to ensure that shortest paths follow the trend of the data. This constraint is shared with existing approaches [Arvanitidis et al., 2018]. Some heuristics exists [Detlefsen et al., 2019b], but principled approaches are currently lacking.

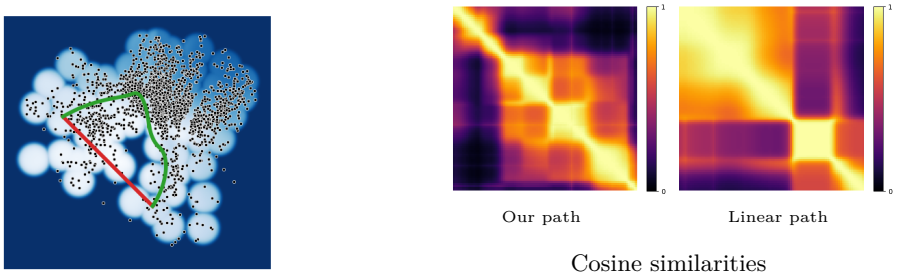


Figure 3.8. Our path (green) follows users with similar preferences, as similarity is only locally high. Instead, the line (red) does not respect the learned structure resulting to users with no specific preferences.

CHAPTER 4

Density estimation on smooth manifolds with normalizing flows

This chapter is adapted from Kalatzis et al. [2021]

Authors: Dimitris Kalatzis, Johan Ziruo Ye, Alison Pouplin, Jesper Wohlert, Søren Hauberg

Abstract

We present a framework for learning probability distributions on topologically non-trivial manifolds, utilizing normalizing flows. Current methods focus on manifolds that are homeomorphic to Euclidean space, enforce strong structural priors on the learned models or use operations that do not easily scale to high dimensions. In contrast, our method learns distributions on a data manifold by "gluing" together multiple local models, thus defining an open cover of the data manifold. We demonstrate the efficiency of our approach on synthetic data of known manifolds, as well as higher dimensional manifolds of unknown topology, where our method exhibits better sample efficiency and competitive or superior performance against baselines in a number of tasks.

4.1 Introduction

Normalizing flows [Papamakarios et al., 2021, Rezende and Mohamed, 2015] provide an elegant framework for modelling complex, multimodal probability distributions. Normalizing flows comprise a base distribution P_U on a latent space U and a diffeomorphism, which provides a 1-to-1 mapping of points from the data space to the latent space according to this base distribution. Given a data point $\boldsymbol{x}$, the marginal likelihood can be computed via the change of variables formula

$p(\boldsymbol{x}) = p(\boldsymbol{u})|\det J_f(\boldsymbol{u})|^{-1} = p(\boldsymbol{u})|\det J_{f^{-1}}(\boldsymbol{x})|$ with $\boldsymbol{x} = f(\boldsymbol{u})$. Typically, the base distribution P_U is a normal or a uniform distribution, both of which are defined in Euclidean space.

Real world data, however, often lie on a manifold, with examples including protein structures [Boomsma et al., 2008, Hamelryck et al., 2006], geological data [Karpatne et al., 2018, Peel et al., 2001] or graph-structured and hierarchical data [Roy et al., 2007, Steyvers and Tenenbaum, 2005]. Diffeomorphisms preserve the topological properties of their domain and therefore modelling the density of manifold-valued data is a known failure mode of flows, due to the topological mismatch between the target distribution $P_{X^\star}$ and the base distribution P_U [Cornish et al., 2020, Dinh et al., 2019, Dupont et al., 2019]. In response, recent works have constructed flows for specific manifolds, such as tori, spheres and hyperbolic spaces [Bose et al., 2020, Rezende et al., 2020].

Still, in many realistic situations one may not know the topological properties of a given data set a priori, but one may reasonably assume an underlying manifold structure. Such cases generally fall under the *manifold hypothesis* [Fefferman et al., 2016], an important heuristic in machine learning, which states that high dimensional data can be described by a low dimensional submanifold embedded in the observation space. Brehmer and Cranmer [2020] propose to learn the shape of the manifold via learning a (single) chart to it, however this implies that the manifold's topological structure is Euclidean. Another set of works [Falorsi and Forré, 2020, Lou et al., 2020, Mathieu and Nickel, 2020, Rozen et al., 2021] exploit local geometric information to learn distributions on embedded submanifolds with non-Euclidean topology but these operations do not easily scale to high dimensions. So the question then emerges: Can flow models learn a probability distribution on manifolds with complex topology and also scale to higher dimensions?

Our approach leverages the class of functions typically learned by flow models to learn a collection of smooth coordinate charts that cover the data manifold. Unlike existing methods, which do not make assumptions on manifold topology, we are able to learn probability distributions on data manifolds with complex (non-Euclidean) topological structure (Fig. 4.1). Furthermore, in contrast to methods that depend on local geometry, our model scales to high dimensional non-Euclidean data. Finally, we are able to achieve competitive or superior performance in all tasks with better sample efficiency and faster runtimes than most of our baselines.

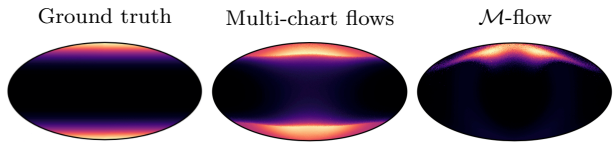


Figure 4.1. A bimodal distribution on a sphere. Contrary to our approach, single-charted models (like the $\mathcal{M}$-flow model [Brehmer and Cranmer, 2020]) struggles to push probability mass to cover both modes.

4.2 A multi-charted approach to density estimation on manifolds

We now present our main contribution, *Multi-chart flows* (MCF). We introduce the construction of density functions on smooth manifolds and subsequently discuss training, inference and the generative process.

4.2.1 Model specification

Given a local coordinate chart (U, ϕ) on the manifold, a probability density p_U supported on a neighborhood $U \subset \mathcal{M}$ can be expressed through the change of variables formula:

$$p_U(\boldsymbol{x}) = p_V(\boldsymbol{u}) |\det G(\boldsymbol{u})|^{-\frac{1}{2}} \tag{4.1}$$

where p_V denotes a simple base density (e.g. a standard Gaussian) over the Euclidean subset V, $\boldsymbol{u} = \phi(\boldsymbol{x})$ and $G = J_{\phi^{-1}}^\top J_{\phi^{-1}}$ is induced by the smooth embedding ϕ^{-1} with the corresponding Jacobian matrix $J_{\phi^{-1}} \in \mathbb{R}^{D \times d}$. Here the more general form of the volume form is used, since ϕ^{-1} is injective. We seek to construct a probability density function $p_{\mathcal{M}} : \mathcal{M} \to \mathbb{R}$ over the manifold, by "gluing" together multiple local models defined in subsets $U \subset \mathcal{M}$. To achieve this we will turn to a partition of unity construction [Lee, 2013, Strichartz, 2003] of such a density function. Let $\{U_i\}_{i=1}^K$ be an open cover of $\mathcal{M}$. Partitions of unity are families $\{f_i\}_{i=1}^K$, of continuous functions $f : \mathcal{M} \to \mathbb{R}$ with $\text{supp} f_i \subseteq U_i$ that satisfy the following:

1. In a neighborhood around a point $\boldsymbol{x} \in \mathcal{M}$, only a finite subset of $\{f_i\}$ are non-zero.
2. $\sum_{i=1}^K f_i(\boldsymbol{x}) = 1$.

As such, we can construct our density function $p_{\mathcal{M}}$ over the manifold by "blending" together the density functions p_U defined in local neighborhoods/coordinate patches on the manifold (eq. 4.1). Thus, with $i = 1, \ldots, K$ denoting the index of the neighborhood and K the number of the overall neighborhoods in our cover of $\mathcal{M}$, which we treat as a hyperparameter we have:

$$p_{\mathcal{M}}(\boldsymbol{x}) = \sum_{i=1}^K w_i p_{U_i}(\boldsymbol{x}) = \sum_{i=1}^K w_i p_{V_i}(\boldsymbol{u}) |\det G_i(\boldsymbol{u})|^{-\frac{1}{2}}, \tag{4.2}$$

where $\sum_{i=1}^K w_i = 1$ and $\boldsymbol{u} = \phi_i(\boldsymbol{x})$. We can furthermore normalize $w_i p_{U_i}(\boldsymbol{x})$ to satisfy the second condition of the partition of unity. This construction is convenient since it

simultaneously allows us to define an open cover over our data manifold, which we can use as a smooth atlas, and removes the need to explicitly learn a reconstruction of the embedded manifold. The overall topological structure is preserved by constructing the manifold from locally Euclidean models. Furthermore, we avoid continuity/differentiability issues at the neighborhood boundaries. Because we are using flow models for our coordinate maps ϕ_i, smooth chart compatibility is ensured by construction, since for overlapping coordinate charts $(U_1, \phi_1), (U_2, \phi_2)$, the composition $\phi_2 \circ \phi_1^{-1}$ is a diffeomorphism as it is a composition of diffeomorphisms.

4.2.2 Introducing a lower bound to the density

While the determinant term in eq. 4.2 can be computed exactly, it involves evaluating $G = J_{\phi_i^{-1}}^{\top} J_{\phi_i^{-1}}$, which is prohibitively expensive even for a modest number of dimensions, since computing the determinant is an $O(d^3)$ operation. We introduce a lower bound to the log likelihood contribution of each chart (eq. 4.1), thereby lower bounding the complete data log likelihood (eq. 4.2). We will replace the determinant with the trace of G which is an $O(d)$ operation. A sketch of a proof follows, with all details in Appendix C.1. We drop neighborhood indices i and for the log likelihood in a given coordinate patch U with coordinate map ϕ, we denote the singular values of $J_{\phi^{-1}}$ by $\{s_i\}_{i=1}^d$ and we have:

$$\log p_U(\boldsymbol{x}) = \log p_V(\boldsymbol{u}) - \frac{1}{2} \log \det |G(\boldsymbol{u})| \tag{4.3}$$

$$= \log p_V(\boldsymbol{u}) - \frac{1}{2} \sum_{i=1}^{d} \log s_i^2. \tag{4.4}$$

Using Jensen's inequality with uniform weights $a_i = 1/d$ we can bound this density by:

$$\log p_U(\boldsymbol{x}) \geq \log p_V(\boldsymbol{u}) - \frac{d}{2} \log \left(\sum_{i=1}^{d} s_i^2 \right) + c \tag{4.5}$$

$$= \log p_V(\boldsymbol{u}) - \frac{d}{2} \log \operatorname{Tr}\left[(J_{\phi^{-1}}(\boldsymbol{u}))^{\top} J_{\phi^{-1}}(\boldsymbol{u}) \right] + c, \tag{4.6}$$

where $c = d \log(d)/2$ is a constant. We can compute the trace efficiently using Hutchinson's estimator [Hutchinson, 1989], arriving at:

$$\log p_U(\boldsymbol{x}) \geq \log p_V(\boldsymbol{u}) - \frac{d}{2} \log \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[||J_{\phi^{-1}}^{\top} \boldsymbol{\epsilon}||_2^2 \right] \tag{4.7}$$

with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I_D)$. Because for all i we have $w_i \in [0, 1]$, inequality 4.9 below holds for all neighborhoods U_i, and by extension the lower bound holds for the overall data

log likelihood on the manifold:

$$\log p_{\mathcal{M}}(\boldsymbol{x}) = \log \sum_{i}^{K} w_i p_{U_i}(\boldsymbol{x}) = \log \sum_{i}^{K} w_i p_{V_i}(\boldsymbol{u}) \det |G_i(\boldsymbol{u})|^{-\frac{1}{2}} \tag{4.8}$$

$$\geq \log \left[C \cdot \sum_{i}^{K} w_i p_{V_i}(\boldsymbol{u}) \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[||J_{\phi^{-1}}^{\top} \boldsymbol{\epsilon}||_2^2 \right]^{-\frac{d}{2}} \right] \qquad \text{with } C = d^{d/2}. \tag{4.9}$$

4.2.3 Training

We train our model using maximum likelihood estimation on the lower bounded density (eq. 4.9). As mentioned before we do not need an explicit manifold learning/reconstruction step. We, furthermore, construct our coordinate maps as embeddings. To construct such a map using flow models, we append $D - d$ zeros to the base variable $\boldsymbol{u} \in \mathbb{R}^d$ and map to $\mathcal{M} \subset \mathbb{R}^D$ with ϕ^{-1}. We denote this "augmented" variable by $\boldsymbol{u}' = [u_1, \ldots, u_d, 0, \ldots, 0]^{\top} \in \mathbb{R}^D$ and for the remainder of the paper we will use this symbol to refer to this construction. As for the forward maps $\phi : U \subset \mathcal{M} \to V \subset \mathbb{R}^d$ we follow the strategy of Beitler et al. [2021], where we split the dimensions of the observed variable $\boldsymbol{x}$ into the intrinsic manifold dimensions d and the directions normal to the manifold $D - d$. We then use the map ϕ to map the manifold dimensions to the base distribution p_V in the Euclidean subset V and the orthogonal directions to a distribution $p_{V^{\perp}}$, which is tightly centered around 0, e.g. a zero-centered Gaussian with $\sigma^2 = 0.01$. This way we can define the map ϕ as a projection from the manifold to the Euclidean domain V_i. Crucially and in contrast to Beitler et al. [2021] we train on the more general form of the change of variables employing the correct volume measure induced by our embeddings ϕ^{-1}, i.e. $dV = \sqrt{G(\boldsymbol{u})} d\boldsymbol{u}$, with G a Riemannian metric tensor defined as: $G = J_{\phi^{-1}}^{\top} J_{\phi^{-1}}$. Although it is flexible, this construction deprives us of the ability to directly compare likelihoods across models, since the volume measure will depend on the chart parameterization of the manifold through the inverse coordinate map ϕ^{-1}.

4.2.4 Sampling

To determine the mixture probabilities for our model we use a neural network during training, in other words $\boldsymbol{w} = \mathrm{NN}(\boldsymbol{x}; \theta)$ with θ denoting the neural network parameters, $\boldsymbol{w}$ a normalized vector in $\mathbb{R}^K$ and K the number of coordinate charts. To be able to sample from our model however, we assume a Categorical distribution over the charts and keep estimates of these probabilities throughout training, by simply normalizing the counts of coordinate chart assignments over the whole dataset. Denoting the index of a data point by $n = 1, \ldots, N$, the index of a coordinate chart by $i = 1, \ldots, K$ and

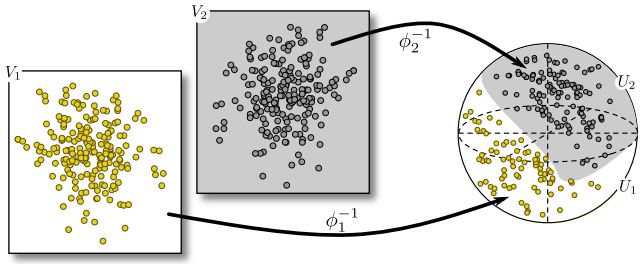


Figure 4.2. Overview of the sampling/generative process proposed in Multi-chart flows. First, the index k of some Euclidean subset V_k is sampled. Then, points sampled in the lower dimensional Euclidean spaces V_k are mapped onto the embedded data manifold by the inverse coordinate maps, ϕ_i^{-1}.

the number of data points assigned to coordinate chart i by N_i:

$$\tilde{w}_i = \frac{N_i}{N} \tag{4.10}$$

$$p(\boldsymbol{c}; \tilde{\boldsymbol{w}}) = \prod_{i=1}^{K} \tilde{w}_i^{\boldsymbol{c}_i} \tag{4.11}$$

Then, we can sample from the model through ancestral sampling, where we first sample chart $\boldsymbol{c}_k \sim p(\boldsymbol{c}; \tilde{\boldsymbol{w}})$, then the d-dimensional latent variable $\boldsymbol{u}$, append $D - d$ zeros to get $\boldsymbol{u}'$ and map to $\mathcal{M} \subset \mathbb{R}^D$ with ϕ_k^{-1} (Fig. 4.2).

4.3 Related work

Learning the manifold structure. Brehmer and Cranmer [2020] propose learning the topological structure of the manifold separately from learning the probability distribution on it and so they split training into two distinct phases. Initially they learn a reconstruction of the data manifold via an embedding $g : \mathcal{M} \to \mathbb{R}^D$, which can be considered a composition $f \circ \phi$ with $\phi : \mathcal{M} \to \mathbb{R}^d$ the manifold chart and $f : \mathbb{R}^d \to \mathbb{R}^D$ a smooth, injective map. Then they learn the density on the manifold via a transformation $h : \mathbb{R}^d \to \mathbb{R}^d$. A crucial limitation is that the data manifold is assumed to be covered by the single chart ϕ, i.e. it is homeomorphic to Euclidean space. The model, thus, cannot represent non-trivial manifolds. Lou et al. [2020] proposed another closely related method treating the exponential map as a chart. Since the exponential map $\exp : T_x\mathcal{M} \to \mathcal{M}$ is a local diffeomorphism between the (Euclidean) tangent space at x and the manifold $\mathcal{M}$, it is treated as a chart ϕ centered at x. They learn a vector field in $T_x\mathcal{M}$ by solving a local ODE for a short time interval, which is mapped onto $\mathcal{M}$ by the expmap. They use the inverse chart, $\log : \mathcal{M} \to T_{x'}\mathcal{M}$, to map to the new tangent space centered at x' and repeat the process. In principle,

this scheme is general, but in practice, the exponential and logarithmic maps are prohibitively expensive for high dimensional manifolds. Rozen et al. [2021] propose Moser flow (MF), a generative model where the learned density consists of a source distribution minus the divergence of a neural network. Therefore, their suggested model falls within the broader family of continuous normalizing flows (CNFs), however they approximate the local divergence operator instead of solving the ODE, achieving significant speedups against CNF-based models (such as e.g. [Grathwohl et al., 2018, Mathieu and Nickel, 2020]) in low dimensions. An important limitation, however, is that the divergence is computationally expensive to approximate in high dimensions, limiting the applicability of MF to general high dimensional settings. Finally, works that learn an atlas of the manifold have appeared in the literature. Nascimento et al. [2014] use Gaussian processes for the chart maps which are combined probabilistically to form an atlas, Pitelis et al. [2013] combine local linear models into an atlas by minimizing a regularized reconstruction error that encourages a small number of charts. Brand [2002] uses a mixture of kernel-based linear projections to build a common coordinate system of connected Euclidean patches. Finally, Schonsheck et al. [2019] propose autoencoder-based coordinate maps to construct their atlas.

Flows on fixed manifolds. A related body of work pertains to flows on manifolds with a priori known topological structure. Rezende et al. [2020] construct flows defined on circles, tori and spheres through projective transformations, as well as by adapting Euclidean models, such as autoregressive flows [Papamakarios et al., 2017] and spline flows [Durkan et al., 2019a, Müller et al., 2019]. Flow-based models defined in hyperbolic space were presented by Bose et al. [2020], wherein two variants are proposed, which use parallel transport of vectors and repeated calls to the exponential and logarithmic maps to map between the tangent bundle and the manifold. A more general method of learning a flow on a manifold was proposed by Gemici et al. [2016], which assumes knowledge of a coordinate chart $\phi : \mathcal{M} \to \mathbb{R}^d$ and an embedding $g : \mathbb{R}^d \to \mathbb{R}^D$ with $d < D$. A limitation here is that $\mathcal{M}$ needs to be homeomorphic to $\mathbb{R}^d$, since it is described by a single chart. When ϕ and g are learned we arrive at the models presented by Brehmer and Cranmer [2020]. We generalize this setting by learning transformations between patches of the manifold $\mathcal{M}$ and subsets of Euclidean space $\mathbb{R}^D$.

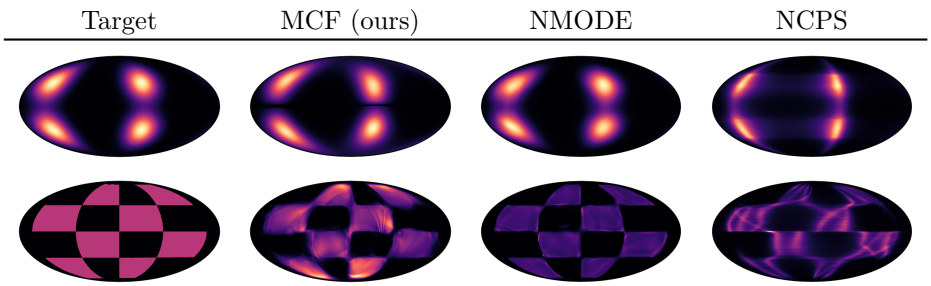


Figure 4.3. Density estimation on the sphere $\mathbb{S}^2$ visualized via the Mollweide projection. Baselines: Neural Manifold ODEs (NMODE) by Lou et al. [2020] and NCPS by Rezende et al. [2020]

4.4 Experiments

4.4.1 Qualitative experiments: Estimation of synthetic densities on 2D manifolds

For our first experiment we trained our model, denoted MCF (Multi-chart flows), on synthetic densities on the sphere $\mathbb{S}^2$, a 2D manifold with well studied topological structure. Our baselines were chosen among models that encode topological information as structural priors by way of a prescribed chart to access the manifold, and models that generally rely on the exponential map which still encodes local topological information on the manifold. Of the former, we chose the recursive circular spline flow (NCPS) [Rezende et al., 2020]. As for the latter, we chose neural manifold ODEs (NMODE) [Lou et al., 2020]. Results can be seen in Figure 4.3. Our approach achieves improved performance over NCPS and performs on par with NMODE at significantly reduced running times (see section 4.4.5). For the "four wrapped normals" dataset (Fig. 4.3 top row) MCF uses two coordinate charts and each coordinate map comprises two rational quadratic (RQ) coupling layers interspersed with LU-decomposed, invertible linear maps. For the "checkerboard" dataset (Fig. 4.3 bottom row), MCF uses four coordinate charts, with each coordinate map comprising three RQ coupling layers interspersed with LU-decomposed linear maps. Complete experimental details can be found in Appendix C.2

4.4.2 Qualitative experiments: Estimation of real world densities on 2D manifolds

We next examine a scenario of real world densities. Our datasets contain the locations of two types of natural disasters: earthquakes [NOAA, 2020] and fires [EOSDIS, 2020]. These distributions are represented on the sphere $\mathbb{S}^2$. Their complexity and multimodality make them suitable test cases for assessing MCF's usefulness in real world scenarios. Figure 4.4 shows the model's results. The density learned by MCF generally captures the modes and patterns in the data and can serve as a modelling tool which can be subject to further refinement by domain experts. As baselines, we trained NCPS and NMODEs, the same models we trained on spherical densities in section 4.4.1, but could not achieve satisfactory results. We include them for completeness along with different spherical projections in Appendix C.3.

4.4.3 Qualitative experiments: Lorenz attractor

Next, we model a distribution residing on a topologically non-trivial manifold. We illustrate our model's ability to preserve the "global" manifold structure and learn the probability density of the Lorenz system by training on points along sampled trajectories. The stable manifold of the system's trajectories is a genus 2 manifold embedded in $\mathbb{R}^3$. For the classical parameter values, the Lorenz attractor admits a Sinai-Ruelle-Bowen (SRB) measure with support over the surface of the system [Tucker, 2002]. Informally, we can say that initial values "diffuse" over this surface.

To create an i.i.d. dataset we generated 100 trajectories using the classical parameter values for the system, then uniformly sampled positions $\boldsymbol{x}(t) \in \mathbb{R}^3$ for $t \in [0, 1000]$ along these. The procedure for the creation of the data set matches that of Brehmer and Cranmer [2020]. Our model consists of flows comprising five layers of RQ coupling transformations and models the manifold using two coordinate charts. More details on architectures and hyperparameter settings can be found in Appendix C.4.

Fig. 4.5 shows the manifold and probability distribution learned by our model and $\mathcal{M}$-flow. Parameterizing the manifold with multiple charts allows MCF to preserve the global topological structure of the manifold and to learn the probability distribution on it, even though the surface is self-intersecting. The single charted $\mathcal{M}$-flow struggles to accurately reconstruct the manifold, as it tries to cover the surface with a single coordinate chart, which implies the surface is homeomorphic to the plane. We do note however that $\mathcal{M}$-flow has captured the coarse-grained topological features of the surface, e.g. the reconstructed manifold is still genus 2 (i.e. contains two "holes").

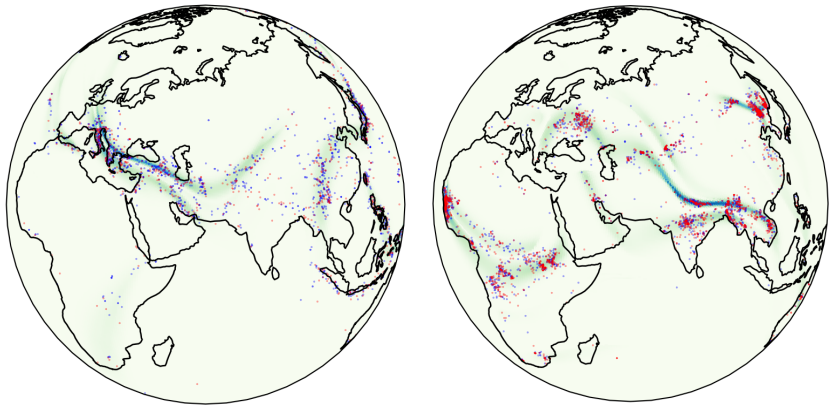


Figure 4.4. Density estimation results on real world densities. Learned density of the *earthquakes* (left) and *fires* (right) geological data as distributions on a sphere. Blue points are sampled from the training set, while red points are sampled from the evaluation set.

4.4.4 Quantitative experiments: Real world particle physics data

Our quantitative experiment focuses on the task of inferring the parameters of a proton-proton collision process at the Large Hadron Collider (LHC). Raw data is usually in the order of millions, but following common practice among domain experts, we use a vector of 40 features to represent the data. The model of the process is based on a simulator which generates data $\boldsymbol{x} \in \mathbb{R}^{40}$ given parameters $\boldsymbol{\theta} \in \mathbb{R}^3$ according to an implicit probability distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$. From domain experts we know that the data resides in a 14-dimensional manifold embedded in $\mathbb{R}^{40}$. Given the observations $\boldsymbol{x}$ and parameters θ, our task is to infer the posterior distribution over the parameters $p(\boldsymbol{\theta}|\boldsymbol{x})$. Thus, we train our model as a conditional density estimator to learn the simulator likelihood function.

Baseline models include a Euclidean flow in the ambient space (RQ-Flow, [Durkan et al., 2019a]), the $\mathcal{M}$-flow model, as well as an $\mathcal{M}$-flow variant with an unrestricted encoder denoted by $\mathcal{M}_e$-flow, both of which were proposed by Brehmer and Cranmer [2020]. Furthermore, Brehmer and Cranmer [2020] introduced versions of the models trained with the SCANDAL method [Brehmer et al., 2020], which improves inference performance. All baselines are composed of thirty-five RQ coupling layers, interspersed with invertible, LU-decomposed linear transformations. The RQ-Flow is trained with maximum likelihood, while the $\mathcal{M}$-flow models are trained in two phases, as in [Brehmer and Cranmer, 2020] corresponding to a manifold learning phase and a density estimation phase. Our model (MCF) comprises five coordinate charts and each coordinate map is composed of ten RQ coupling layers, interspersed with invertible, LU-decomposed linear transformations. For further details on architectures and

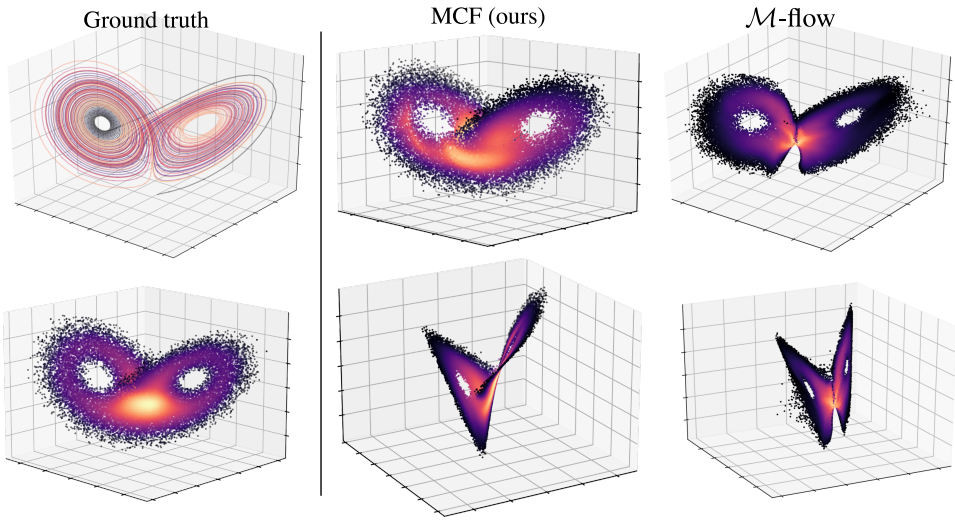


Figure 4.5. The manifold and probability distribution for the Lorenz attractor system. For the $\mathcal{M}$-flow model, the depicted manifold is learned, whereas for MCF the manifold shape is implicitly preserved through locally invertible models. Brighter color represents areas of higher estimated density. Ground truth shows sampled trajectories and the implicit surface formed by the system.

hyperparameters, see Appendix C.5.

Model	sample closure ↓	log posterior ↑
RQ-Flow [Durkan et al., 2019a]	**0.0019** ± 0.0001	-3.94 ± 0.87
RQ-Flow (SCANDAL)	0.0565 ± 0.0059	-0.49 ± 0.09
$\mathcal{M}$-flow [Brehmer and Cranmer, 2020]	0.0045 ± 0.0004	-1.71 ± 0.30
$\mathcal{M}$-flow (SCANDAL)	0.0045 ± 0.0004	0.11 ± 0.04
$\mathcal{M}_e$-flow [Brehmer and Cranmer, 2020]	0.0046 ± 0.0002	-1.44 ± 0.34
$\mathcal{M}_e$-flow (SCANDAL)	0.0291 ± 0.0010	0.03 ± 0.09
MCF [ours]	0.0040 ± 0.001	**0.55** ± 0.21

Table 4.1. Quantitative results on the large hadron collider (LHC) data. *Sample closure* measures sample quality (lower is better), log-posterior score $\log p(\boldsymbol{\theta}|\boldsymbol{x}_{\text{obs}})$ measures quality of inference (higher is better). Each model is trained five times with independent initializations. The top and bottom values are removed and the mean is computed over the remaining runs. Best results are shown in bold. Baseline results by Brehmer and Cranmer [2020].

For model evaluation, first we investigate the generative capabilities of all models by evaluating a series of tests on model samples. These "closure tests" are a weighted sum of individual constraints encoding relationships (derived from domain knowledge) between dimensions in the observed vector, taking values in $[0, 1]$, where smaller values

denote higher sample quality. Second, we measure the quality of the log posterior inference. Given a set of 20 observed samples $\boldsymbol{x}_{\text{obs}} \sim p(\boldsymbol{x}|\boldsymbol{\theta}^\star)$, we evaluate model likelihood in an MCMC sampler to generate posterior samples $\theta \sim p(\boldsymbol{\theta}|\boldsymbol{x}_{\text{obs}})$. To evaluate the posterior, we then use kernel density estimation with a Gaussian kernel. We evaluate all models for three different ground truth parameter points $\boldsymbol{\theta}^\star$. For more details on the experimental setting of the task, see Brehmer and Cranmer [2020].

Table 4.1 summarizes results for the LHC data. While the RQ-Flow learns a good sampler for the observed data judging by the closure test score, it does not estimate the density well, as evidenced by the log posterior score. Maximum likelihood in the ambient space does not take manifold topology into account, rather it relies on models with enough capacity to map the data to a base distribution in the ambient space. To the extent the model manages to learn such a mapping, it will be an adequate data sampler but will concurrently lead to biased density estimates due to the mismatch in the volume measures. Conversely, models that learn (such as $\mathcal{M}$-flow) or preserve (such as MCF) the topological structure of the data, achieve more accurate density estimates. Using multiple charts, our method outperforms all baselines in log posterior scores. In terms of sample quality, our method yields marginally better results than the single-charted baselines, which could imply that the underlying manifold is homeomorphic to Euclidean space, meaning a single chart is enough to capture its topology, however using multiple charts is beneficial for density estimation.

4.4.5 Running times

Respecting the topology of the data manifold yields tangible benefits to runtimes. Our approach generally uses fewer flow layers and parameters than most baselines leading to consistently smaller convergence times than all other baselines. Table 4.2 shows model wallclock times. For the experiments on section 4.4.1 all models were trained on the CPU. For all other experiments all models were trained on a Titan X (Pascal) GPU.

4.5 Conclusion

We have presented a flow-based framework for modelling data distributions on non-Euclidean manifolds. Recent works in this direction either encode the topology of the target manifold in the model's architecture, rely on operations that do not scale to high dimensions or can, in principle, only learn Euclidean manifolds. In contrast, our method can generalize to manifolds of higher dimensions and/or complex topology. Our approach can converge faster and to better optima compared to most baselines. Shorter convergence times are not surprising since our approach does not require a lot

Datasets	Models		
	MCF (ours)	NMODE	NCPS
Wrapped normals ($\mathbb{S}^2$)	**1.53** $\pm$0.64	54.28 ± 4.01	2.69 ± 0.42
Checkerboard ($\mathbb{S}^2$)	**3.40** $\pm$0.22	50.81 ± 2.74	8.13 ± 0.82
	MCF (ours)	$\mathcal{M}$-flow	
Lorenz attractor	**16.35** $\pm$0.28	35.84 ± 0.91	
	MCF (ours)	$\mathcal{M}$-flow	RQ-Flow
Large Hadron Collider	**80.5** $\pm$1.32	96.61 ± 1.93	99.74 ± 4.71

Table 4.2. Model wallclock time per dataset (in hours). Computed over 3 training runs.

of capacity to learn subsets of the manifold with simpler topology. ODE-based models and models that exploit local geometry match or surpass the performance of our approach, since geometric operations respect manifold topology, therefore providing a strong inductive bias. However, these models are inherently at a disadvantage regarding computational cost and scalability, since they rely either on sequential solvers, not taking full advantage of parallelization or on approximations of local operators that become prohibitively expensive in higher dimensions. Against models with structural priors, our model converges faster and achieves better optima since it does not rely on classical projective maps (like the cylindrical projection used by Rezende et al. [2020]) which do not preserve topology.

Limitations. Optimizing multi-charted manifold flow models is consistently harder than their Euclidean counterparts, making hyperparameter configuration an important consideration. We also assume that the data resides on a smooth manifold, which might not necessarily be true. Finally, quantitative comparisons with other models become harder as different chart parameterizations of manifolds result in different units for the estimated log-likelihood.

Conclusion

In machine learning and related fields, data is represented as a set of vectors such that they can be efficiently processed by numerical algorithms. While a naïve view of this arbitrary choice could lead to the conclusion that Euclidean models and the related algorithms are well suited to handling such data, we often find that the vector/Euclidean representation does not actually imply a Euclidean underlying structure, in terms of topology and/or geometry. As a result, Euclidean models are either inefficient or fail outright to model such datasets. In other cases we may even want to represent data points as points on a manifold, albeit still with reference to an ambient coordinate system in a vector format. Such examples include hierarchical data, typically represented as points on hyperbolic spaces or amino acids typically represented as points on tori. In such cases Euclidean models are dropped entirely in favor of models developed specifically for these particular spaces.

In this thesis we have presented our approach for designing models that respect the topological and geometric features of the data without making any further assumptions regarding these other than that they are non-Euclidean. In chapter 2 we used the notions of pull-back metrics and geodesics, i.e. the generalization of shortest path distances to curved spaces as building blocks to design a Brownian motion-based prior distribution defined on manifolds as well as endow the latent space with a non-Euclidean geometric structure. We showed how employing this prior preserves the relationships between encoded data points, at least from a perspective of shortest distances between them and alleviates the problems of mapping datasets with compact support onto a Euclidean space under the Gaussian distribution. Whereas in chapter 2 we used Gaussian decoders to facilitate our approach, in chapter 3 we generalized our method to non-Gaussian decoders by considering the manifold of probability distributions and pulling back the related Fisher-Rao metric to the latent space, allowing us to endow the latent space of VAEs with non-reparameterizable and/or discrete decoders with a meaningful geometry. Finally, in chapter 4 we presented our approach for using normalizing flows to form a smooth atlas over the data manifold and learn probability distributions defined on manifolds with complex (i.e. non-Euclidean) topological properties.

By now it should be apparent to the reader that the running theme of our work is finding ways to use elements from the theory of geometry and topology of non-Euclidean spaces to build models that benefit from such considerations. We have presented

results that showcase that the use of such models for exploratory analyses of datasets yields much more sensible outcomes, such as geodesic (as opposed to Euclidean shortest path) interpolations. We judge this to be a strong indication that respecting the geometry of the data manifold preserves relationships between data points, insofar one accepts the argument that these are reflected in the relevant geometry. This approach also yields promising results in downstream tasks that benefit from representations that more accurately preserve these relationships, such as classification. We have also presented a strategy to extend this approach to non-reparameterizable distributions through the use of information geometry, extending its utility beyond the usual optimization setting where its application is related to natural gradients. Importantly, both in the case of VAEs and normalizing flows, the use of non-Euclidean mathematical frameworks represents a general strategy to avoid numerical problems and unstable optimization related to the mismatch between the topology of the data manifold and that of the latent spaces of these models. Furthermore, and for flows in particular, we have shown how to use such Euclidean tools to learn distributions on non-Euclidean spaces. Well performing Riemannian normalizing flow-based approaches that encode the topology or geometry of particular manifolds as structural priors exist in the literature, yet our approach based on forming a smooth atlas over the data manifold and as such, avoiding costly and largely non-scalable Riemannian computations yields faster training times with comparable results.

Future work Models with geometric inductive biases are gaining more traction in the machine learning community, yet scalable models with geometric inductive biases that are general enough as to be agnostic to the particular topological features of the data manifolds in question are still rare. The dichotomy of mathematical principle on the one hand, and intensive tinkering through experimentation on the other is something the machine learning community is no stranger to and with the advent of large scale diffusion-based models, the scales seem tipped in favor of the latter. However, we think that these two approaches need not be antithetical but rather can be made to work in synergy. We consider the models we have presented in this thesis to be the initial steps towards the development of efficient and scalable non-Euclidean tools with more general topological/geometric inductive biases. We believe that such approaches should, at minimum, yield comparable results to large scale Euclidean models, while at the same time be characterized by more efficient training with regards to optimization, training times and sample efficiency. Some initial encouraging results in this direction are already present in this thesis. Thus, there remains a lot of room for future work in merging topological/geometric approaches with large scale models in a way that enhances the efficiency and/or performance of the latter.

Appendices

APPENDIX A

Appendix to chapter 2

A.1 On neural network-based immersions

For the decoder map 2.2 to be a valid immersion, its differential df needs to be injective for all $p \in \mathcal{M}$ as stated in definition 5. The differential of f is represented by its Jacobian matrix J_f and for it to be injective for all $p \in \mathcal{M}$, it needs to be full rank. This is ensured if for the MLPs representing the decoder $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\sigma}_\psi$ the following are true:

- Each hidden layer in the network has an equal or greater number of units to the previous layer ($n_{L-1} \leq n_L$).
- All weight matrices in the network are full rank.
- The activation functions are at least twice differentiable and strictly monotonic.

In our experiments, we opt for the same number of units in each hidden layer of the network and ELU non-linearities. In theory, the ELU activation function could present problems since it has a point of discontinuity at 0, however we did not experience any numerical instability that would arise in such case. All weight matrices are initialized uniformly He et al. [2015] which practically has zero probability of yielding low rank weight matrices. While theoretically this could change via the gradient updates of the weights, this would once again immediately break experiments because of numerical instabilities, which we did not observe.

A.2 Geodesic estimation

We estimate geodesic distances by minimizing curve energy. In detail, we represent the geodesic curve with a cubic spline with parameters initialized to form a straight line. These parameters are then optimized via gradient descent by minimizing the

curve energy:

$$\begin{aligned}\mathcal{E}(\gamma) &= \frac{1}{2}\int_0^1 ||\dot{\gamma}(t)||_g^2 \, dt \\ &= \frac{1}{2}\int_0^1 \dot{\gamma}^\top(t) G_\gamma \dot{\gamma}(t) \, dt \end{aligned} \tag{A.1}$$

where γ is the geodesic curve, $\dot{\gamma}$ is the first derivative of the curve, i.e. its velocity vector and G_γ is the matrix representation of the metric tensor evaluated at the curve points. The integral A.1 is computed by numerical approximation, where the partition of the interval can be chosen as a hyperparameter.

A.3 Experimental setup

The architectures of all model variants are shown below in Tables A.1 and A.2. The encoder mean and variance, as well as the decoder mean are modelled by 2-layer MLPs as shown below. The decoder mean mirrors the encoder mean, while the *precision* $\boldsymbol{\beta}$ is estimated by the RBF network. The number of the RBF centers is set to 350 and the bandwidth is set to 0.01 in all cases. For a fair comparison, all models share the same underlying architecture for the encoder and decoder. Tables A.1 and A.2 summarize the architectures, listing the activation function for each layer with the units corresponding to each layer in parentheses.

Table A.1. Encoder network architectures.

Network	Layer 1	Layer 2	Output
$\boldsymbol{\mu_\phi(x)}$	ELU (300)	ELU (300)	Linear ($dim(\boldsymbol{z})$)
$\boldsymbol{\sigma^2_\phi(x)}$	ELU (300)	ELU (300)	Softplus ($dim(\boldsymbol{z})$)

Table A.2. Decoder network architectures. * denotes strictly positive weights.

Network	Layer 1	Layer 2	Output
$\boldsymbol{\mu_\theta(z)}$	ELU (300)	ELU (300)	Linear ($dim(\boldsymbol{x})$)
$\boldsymbol{\beta_\psi(z)}$	RBF ($\mathbb{R}^{dim(\mathcal{Z})\times 350}$)	Linear* ($dim(\boldsymbol{x})$)	Identity ($dim(\boldsymbol{x})$)

A.3.1 Section 5.1 experiment

Detlefsen et al. [2019a] highlighted the importance of optimizing the mean and variance components separately, when training VAEs with Gaussian generative models.

Following this paradigm, in all our experiments we first optimize the encoder components ($\boldsymbol{\mu_\phi}$ and $\boldsymbol{\sigma_\phi}$) along with the decoder $\boldsymbol{\mu_\theta}$. Then, keeping these fixed, we optimize the decoder $\boldsymbol{\sigma_\psi}$. All models were trained for 300 epochs. More specifically, the $\mathcal{R}$-VAE was trained as an autoencoder (optimizing only the encoder $\boldsymbol{\mu_\phi}$ and $\boldsymbol{\sigma_\phi}$ and the decoder $\boldsymbol{\mu_\theta}$) for the first 100 epochs and for the remaining 200 epochs the latent prior and the decoder $\boldsymbol{\beta_\psi}$ were optimized. Similarly for a VAE, it was deterministically warmed up for 100 epochs and for the remaining 200 epochs, the decoder $\boldsymbol{\beta_\psi}$ was optimized. All experiments were run with the Adam optimizer Kingma and Ba [2015] with default parameter settings and a fixed learning rate of 10^{-3}. The batch size was 100 for all models.

A.3.2 Section 5.2 experiment

The classifier used on this section was a single, 100-unit layer MLP with ReLU nonlinearities, trained for 100 epochs with the Adam optimizer with default parameter settings and a learning rate of 10^{-3}. The batch size was set at 64. The architectures of the models giving rise to the latent representations are as in the previous section.

A.3.3 Runtime comparisons

Below is the wall clock time for every model used in the experiments. The statistics were computed without a fixed seed. The latent space dimensions are denoted by d. In VAE-VampPrior, n denotes the number of mixture components in the latent prior.

Table A.3. Per epoch training time for each model. Mean and std deviation in seconds, computed over 100 epochs on MNIST.

Model	$d = 2$	$d = 5$	$d = 10$
VAE	$10.64_{\pm.51}$	$11.01_{\pm.77}$	$11.10_{\pm.60}$
VAE-VampPrior ($n = 128$)	$10.66_{\pm.6}$	$11.22_{\pm.9}$	$11.37_{\pm.96}$
VAE-VampPrior ($n = 256$)	$10.72_{\pm.3}$	$11.34_{\pm1.21}$	$11.52_{\pm.77}$
VAE-VampPrior ($n = 512$)	$10.9_{\pm.34}$	$11.38_{\pm.93}$	$12.18_{\pm1.12}$
$\mathcal{R}$-VAE	$55.73_{\pm4.36}$	$59.97_{\pm1.33}$	$60.13_{\pm1.19}$

A.3.4 Complete results for VAE-VampPrior

Tables 2.1 & 2.2 show the results of the best performing VampPrior model variant. Here we show the complete results of the VAE-VampPrior in all settings. Below n

denotes the number of mixture components in the latent prior, while d denotes the latent space dimensions.

Table A.4. MNIST results of VAE with VampPrior for varying latent space dimensions and number of mixture components in the latent prior.

Model	Neg. ELBO	Rec	KL
	$d = 2$		
VAE-VampPrior ($n = 128$)	-1039.66$_{\pm 2.56}$	-1042.13$_{\pm 2.56}$	2.46$_{\pm .01}$
VAE-VampPrior ($n = 256$)	-1045.04$_{\pm 5.20}$	-1047.34$_{\pm 5.22}$	2.30$_{\pm .03}$
VAE-VampPrior ($n = 512$)	-1040.79$_{\pm 9.23}$	-1043.24$_{\pm 9.25}$	2.45$_{\pm .05}$
	$d = 5$		
VAE-VampPrior ($n = 128$)	-1100.77$_{\pm 4.98}$	-1102.46$_{\pm 4.91}$	1.69$_{\pm .06}$
VAE-VampPrior ($n = 256$)	-1103.29$_{\pm 1.85}$	-1105.04$_{\pm 1.79}$	1.75$_{\pm .12}$
VAE-VampPrior ($n = 512$)	-1109.74$_{\pm 4.87}$	-1111.63$_{\pm 4.87}$	1.88$_{\pm .01}$
	$d = 10$		
VAE-VampPrior ($n = 128$)	-1110.05$_{\pm 6.10}$	-1112.23$_{\pm 5.82}$	1.84$_{\pm .04}$
VAE-VampPrior ($n = 256$)	-1116.58$_{\pm 4.23}$	-1118.28$_{\pm 4.20}$	1.69$_{\pm .02}$
VAE-VampPrior ($n = 512$)	-1100.64$_{\pm 2.93}$	-1102.42$_{\pm 2.97}$	1.78$_{\pm .03}$

Table A.5. FashionMNIST results of VAE with VampPrior for varying latent space dimensions and number of mixture components in the latent prior.

Model	Neg. ELBO	Rec	KL
	$d = 2$		
VAE-VampPrior ($n = 128$)	-694.63$_{\pm 8.65}$	-697.14$_{\pm 8.65}$	2.50$_{\pm .01}$
VAE-VampPrior ($n = 256$)	-702.67$_{\pm 17.45}$	-705.19$_{\pm 17.44}$	2.52$_{\pm .04}$
VAE-VampPrior ($n = 512$)	-705.90$_{\pm 21.29}$	-708.45$_{\pm 21.29}$	2.54$_{\pm .01}$
	$d = 5$		
VAE-VampPrior ($n = 128$)	-755.80$_{\pm .66}$	-756.58$_{\pm .71}$	0.77$_{\pm .06}$
VAE-VampPrior ($n = 256$)	-767.54$_{\pm 3.22}$	-768.33$_{\pm 3.31}$	0.78$_{\pm .09}$
VAE-VampPrior ($n = 512$)	-769.27$_{\pm 5.0}$	-770.10$_{\pm 5.02}$	0.83$_{\pm .09}$
	$d = 10$		
VAE-VampPrior ($n = 128$)	-754.47$_{\pm 6.78}$	-758.20$_{\pm 6.72}$	3.72$_{\pm .06}$
VAE-VampPrior ($n = 256$)	-756.13$_{\pm 5.40}$	-760.49$_{\pm 5.1}$	3.69$_{\pm .06}$
VAE-VampPrior ($n = 512$)	-774.17$_{\pm 10.83}$	-777.75$_{\pm 10.78}$	3.58$_{\pm .06}$

APPENDIX B

Appendix to chapter 3

B.1 Additional details for information geometry

In this section we provide additional information regarding information geometry. We note that many of these proposition are already know in the literature, however, we include them for completion and for the paper to be standalone.

The Fisher-Rao metric is positive definite only if it is non-singular, and then, defines a Riemannian metric [Nielsen, 2020]. In this paper, we assume that the observation $\boldsymbol{x} \in \mathcal{X}$ is a random variable following a probability distribution $p(\boldsymbol{x})$ such that $\boldsymbol{x} \sim p(\boldsymbol{x}|\eta)$, and any smooth changes of the parameter η would alter the observation $\boldsymbol{x}$. This way, the Fisher-Rao metric used in our paper is non-singular and the statistical manifold $\mathcal{H}$ is a Riemannian manifold.

A known result in *information geometry* [Amari, 2016, Nielsen, 2020] is that the Fisher-Rao metric is the first order approximation of the KL-divergence, as recall in Proposition 5. Using this fact, we can define the Fisher-Rao distance and energy in function of the KL-divergence, leading to Proposition 6.

Proposition 5. *The Fisher-Rao metric is the first order approximation of the KL-divergence between perturbed distributions:*

$$D_{\mathrm{KL}}(p(\boldsymbol{x}|\eta)||p(\boldsymbol{x}|\eta+\delta\eta)) = \frac{1}{2}\delta\eta^{\top} I_{\mathcal{H}}(\eta)\delta\eta + O(\delta\eta^2),$$

with $I_{\mathcal{H}}(\eta) = \int p(\boldsymbol{x}|\eta)\left[\nabla_\eta \log p(\boldsymbol{x}|\eta)\nabla_\eta \log p(\boldsymbol{x}|\eta)^{\top}\right] d\boldsymbol{x}$.

Proof. Let's decompose $\log p(\boldsymbol{x}|\eta+\delta\eta)$ using the Taylor expansion:

$$\log p(\boldsymbol{x}|\eta+\delta\eta) = \log p(\boldsymbol{x}|\eta) + \nabla_\eta \log p(\boldsymbol{x}|\eta)^{\top}\delta\eta + \frac{1}{2}\delta\eta^{\top}\mathrm{Hess}_\eta\left[\log p(\boldsymbol{x}|\eta)\right]\delta\eta + O(\delta\eta^2),$$

where the Hessian is $\mathrm{Hess}_\eta\left[\log p(\boldsymbol{x}|\eta)\right] = \frac{\mathrm{Hess}_\eta[p(\boldsymbol{x}|\eta)]}{p(\boldsymbol{x}|\eta)} - \nabla_\eta \log p(\boldsymbol{x}|\eta)\nabla_\eta \log p(\boldsymbol{x}|\eta)^{\top}$ and the $\nabla_\eta \log p(\boldsymbol{x}|\eta) = \frac{\nabla_\eta p(\boldsymbol{x}|\eta)}{p(\boldsymbol{x}|\eta)}$.

Also $\int \nabla_\eta p(\boldsymbol{x}|\eta)d\boldsymbol{x} = \nabla_\eta \int p(\boldsymbol{x}|\eta)d\boldsymbol{x} = 0$ and $\int \mathrm{Hess}_\eta[p(\boldsymbol{x}|\eta)]d\boldsymbol{x} = \mathrm{Hess}_\eta[\int p(\boldsymbol{x}|\eta)d\boldsymbol{x}] = 0$.

Replacing all those expressions to the first equation finally gives:

$$
\begin{aligned}
&D_{\mathrm{KL}}(p(\boldsymbol{x}|\eta)||p(\boldsymbol{x}|\eta+\delta\eta)) \\
&\qquad = \int p(\boldsymbol{x}|\eta)\log p(\boldsymbol{x}|\eta)d\boldsymbol{x} - \int p(\boldsymbol{x}|\eta)\log p(\boldsymbol{x}|\eta+\delta\eta)d\boldsymbol{x} \\
&\qquad = -\int p(\boldsymbol{x}|\eta)\left(\nabla_\eta \log p(\boldsymbol{x}|\eta)^\top \delta\eta + \frac{1}{2}\delta\eta^\top \mathrm{Hess}_\eta[\log p(\boldsymbol{x}|\eta)]\delta\eta + O(\delta\eta^2)\right) d\boldsymbol{x} \\
&\qquad = \frac{1}{2}\delta\eta^\top \left[\int p(\boldsymbol{x}|\eta)\left[\nabla_\eta \log p(\boldsymbol{x}|\eta)\nabla_\eta \log p(\boldsymbol{x}|\eta)^\top\right] d\boldsymbol{x}\right]\delta\eta + O(\delta\eta^2).
\end{aligned}
$$

□

Definition 8. *We consider a curve $\gamma(t)$ and its derivative $\dot{\gamma}(t)$ on the statistical manifold such that, $\forall t \in [0,1], \gamma(t) = \eta_t \in \mathcal{H}$. The manifold is equipped with the Fisher-Rao metric. The length and the energy functionals are defined with respect to the metric $\mathbf{I}_{\mathcal{H}}(\eta)$:*

$$
\mathrm{Length}(\gamma) = \int_0^1 \sqrt{\dot{\gamma}(t)^\top I_{\mathcal{H}}(\eta)\dot{\gamma}(t)}dt \quad \textit{and} \quad \mathrm{Energy}(\gamma) = \int_0^1 \dot{\gamma}(t)^\top I_{\mathcal{H}}(\eta)\dot{\gamma}(t)dt.
$$

Locally length-minimising curves between two connecting points are called geodesics. These can be found by minimizing the energy using the Euler-Lagrange equations which gives the following system of 2^{nd} order nonlinear ordinary differential equations (ODEs) [Arvanitidis et al., 2018]

$$
\ddot{\gamma}(t) = -\frac{1}{2}I_{\mathcal{H}}^{-1}(\gamma(t))\Big[2(\dot{\gamma}(t)^\top \otimes \mathbb{I}_d)\frac{\partial vec[I_{\mathcal{H}}(\gamma(t))]}{\partial\gamma(t)}\dot{\gamma}(t) - \frac{\partial vec[I_{\mathcal{H}}(\gamma(t))]}{\partial\gamma(t)}^\top (\dot{\gamma}(t)\otimes\dot{\gamma}(t))\Big]. \tag{B.1}
$$

Proposition 6. *The KL-divergence between two close elements of the curve γ is defined as: $KL(p_t,\ p_{t+\delta t}) = D_{\mathrm{KL}}(p(\boldsymbol{x}|\gamma(t))||p(\boldsymbol{x}|\gamma(t+\delta t)))$. The length and the energy functionals can be approximated with respect to this KL-divergence:*

$$
\textit{Length}(\gamma) \approx \sqrt{2\sum_{t=1}^{T} KL(p_t, p_{t+\delta t})} \quad \textit{and} \quad \textit{Energy}(\gamma) \approx \frac{2}{\delta t}\sum_{t=1}^{T} KL(p_t, p_{t+\delta t})
$$

Proof. On the statistical manifold, we have $\gamma(t+\delta t) = \gamma(t) + \delta t\dot{\gamma}(t)$. The KL divergence between perturbed distributions can be defined as:

$$
\mathrm{KL}(p_t, p_{t+\delta t}) = \mathrm{KL}(p(\boldsymbol{x}|\gamma(t)), p(\boldsymbol{x}|\gamma(t+\delta t))) = \mathrm{KL}(p(\boldsymbol{x}|\eta_t), p(\boldsymbol{x}|\eta_t+\delta\eta_t)), \tag{B.2}
$$

with $\eta_t = \gamma(t)$ and $\delta\eta_t = \delta t\ \dot{\gamma}(t)$. Then, we obtain:

$$
\mathrm{KL}(p_t, p_{t+\delta t}) = \frac{1}{2}\delta t^2\ \dot{\gamma}(t)^\top \mathbf{I}_{\mathcal{H}}(\eta_t)\dot{\gamma}(t) + O(\delta t^2).
$$

The length and energy terms appear in the following equations:

$$\int_0^1 \mathrm{KL}(p_t, p_{t+\delta t})dt = \frac{\delta t^2}{2}\int_0^1 \dot\gamma(t)^\top \mathbf{I}_{\mathcal{H}}(\eta_t)\dot\gamma(t)dt + O(\delta t^2) = \frac{\delta t^2}{2}\,\mathrm{Energy}(\gamma) + O(\delta t^2),$$

$$\int_0^1 \sqrt{\mathrm{KL}(p_t, p_{t+\delta t})}dt = \frac{\delta t}{\sqrt{2}}\int_0^1 \sqrt{\dot\gamma(t)^\top \mathbf{I}_{\mathcal{H}}(\eta_t)\dot\gamma(t)}dt + O\delta t^2) = \frac{\delta t}{\sqrt{2}}\,\mathrm{Length}(\gamma) + O(\delta t^2).$$

If we want approximate any continuous function f with a discrete sequence, by partitioning it in T small segments, such that: $\delta t \approx \frac{1}{T}$, we have: $\int_0^1 f(t)dt \approx \sum_{t=1}^T f(t)\delta t$, which in our case gives:

$$\mathrm{Length}(\gamma) \approx \sqrt{2\sum_{t=1}^T \mathrm{KL}(p_t, p_{t+\delta t})} \quad \text{and} \quad \mathrm{Energy}(\gamma) \approx \frac{2}{\delta t}\sum_{t=1}^T \mathrm{KL}(p_t, p_{t+\delta t}).$$

□

B.1.1 The Fisher-Rao metric for several distributions

Distributions	PDFs	Parameters	Fisher-Rao matrix
Normal	$\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{(\boldsymbol{x}-\mu)^2}{2\sigma^2}\right\}$	μ, σ^2	$\mathbf{I}_{\mathcal{N}}(\mu, \sigma^2)$
Bernoulli	$\theta^{\boldsymbol{x}}(1-\theta)^{1-\boldsymbol{x}}$	θ	$\mathbf{I}_{\mathcal{B}}(\theta)$
Categorical	$\prod_{k=1}^K \theta_k^{\boldsymbol{x}_k}$	$\theta_1, \ldots, \theta_K$	$\mathbf{I}_{\mathcal{C}}(\theta_1, \ldots, \theta_K)$
Gamma	$\frac{\beta^\alpha \boldsymbol{x}^{\alpha-1}e^{-\beta\boldsymbol{x}}}{\Gamma(\alpha)}$	α, β	$\mathbf{I}_{\mathcal{G}}(\alpha, \beta)$
Von Mises-Fisher ($\mathbb{S}^2$)	$\frac{\kappa}{4\pi\sinh\kappa}\exp\left\{(\kappa\mu^\top \boldsymbol{x})\right\}$	κ, μ	$\mathbf{I}_{\mathcal{S}}(\kappa, \mu)$
Beta	$\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}\boldsymbol{x}^{\alpha-1}(1-\boldsymbol{x})^{\beta-1}$	α, β	$\mathbf{I}_{\mathcal{B}}(\alpha, \beta)$

Table B.1. List of distributions

With the notations of Table B.1, the Fisher-Rao matrices of the the univariate Normal, Bernoulli and Categorical are:

$$\mathbf{I}_{\mathcal{N}}(\mu, \sigma^2) = \begin{pmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{2\sigma^2} \end{pmatrix}, \quad \mathbf{I}_{\mathcal{B}}(\theta) = \frac{1}{\theta(1-\theta)}, \quad \mathbf{I}_{\mathcal{C}}(\theta_1, \ldots, \theta_K) = \mathrm{diag}(1/\theta_1, \ldots, 1/\theta_K)$$

In addition, the Fisher-Rao matrices of the Gamma, Von Mises-Fisher and the Beta distributions are:

$$\mathbf{I}_{\mathcal{G}}(\alpha,\beta) = \begin{pmatrix} \frac{\alpha}{\beta^2} & -\frac{1}{\beta} \\ -\frac{1}{\beta} & \Psi_1(\alpha) \end{pmatrix},$$
$$\mathbf{I}_{\mathcal{S}}(\kappa,\mu) = \begin{pmatrix} \kappa K(\kappa)(\mathbb{1} - 3\mu\mu^\top) + \kappa^2\mu\mu^\top & (\kappa K(\kappa)^2 - \frac{2}{k}K(\kappa) + 1)\mu \\ (\kappa K(\kappa)^2 - \frac{2}{k}K(\kappa) + 1)\mu^\top & 3K(\kappa)^2 - \frac{2}{\kappa}K(\kappa) + 1 \end{pmatrix},$$
$$\mathbf{I}_{\mathcal{B}}(\alpha,\beta) = \begin{pmatrix} \Psi_1(\alpha) - \Psi_1(\alpha+\beta) & -\Psi_1(\alpha+\beta) \\ -\Psi_1(\alpha+\beta) & \Psi_1(\beta) - \Psi_1(\alpha+\beta) \end{pmatrix},$$

with $\Psi_1(\alpha) = \frac{\partial^2 \ln\Gamma(\alpha)}{\partial\alpha}$ the trigamma function, and $K(\kappa) = \coth\kappa - \frac{1}{\kappa}$.

Proof. The univariate Normal, Bernoulli and Categorical have already been studied by Tomczak [2012], and the Beta distribution by Brigant and Puechmorel [2019]. We will then focus our proof on the Gamma and the Von-Mises Fisher distributions.

In order to bypass unnecessary details, we will use the following notations, we redefine the Fisher-Rao as: $\mathbf{I}(\eta) = \mathbb{E}_x[g(\eta,x)g(\eta,x)^\top]$, with $g(\eta,x) = \nabla_\eta \ln p(x|\eta)$ the Fisher score. We call $G = g(\eta,x)g(\eta,x)^\top$, and G_{ij} the matrix elements.

Gamma distribution:

We have $p(x|\alpha,\beta) = \Gamma(\alpha)^{-1}\beta^\alpha x^{\alpha-1}e^{-\beta x}$, which leads to:

$$\begin{aligned} \ln p(x|\alpha,\beta) &= -\ln\Gamma(\alpha) + \alpha\ln\beta + (\alpha-1)\ln x - \beta x, \\ \frac{\partial \ln p}{\partial\alpha} &= -\Psi(\alpha) + \ln\beta + \ln x, \\ \frac{\partial \ln p}{\partial\beta} &= \frac{\alpha}{\beta} - x. \end{aligned}$$

Then:

$$\begin{aligned} G_{11} &= \left(\frac{\partial \ln p}{\partial\alpha}\right)^2 = (\Psi_0(\alpha) + \ln\beta)^2 + 2(\Psi(\alpha) + \ln\beta)lnx + \ln^2 x, \\ G_{22} &= \left(\frac{\partial \ln p}{\partial\beta}\right)^2 = \left(\frac{\alpha}{\beta}\right)^2 - 2\frac{\alpha}{\beta}x + x^2, \\ G_{12} = G_{21} &= \frac{\partial \ln p}{\partial\alpha}\cdot\frac{\partial \ln p}{\partial\beta} = (\Psi(\alpha) + \ln\beta)\left(\frac{\alpha}{\beta} - x\right) + \frac{\alpha}{\beta}\ln x - x\ln x. \end{aligned}$$

We know that $\mathbb{E}[x] = \frac{\alpha}{\beta}$. We can compute, using your favorite symbolic computation

software, the following moments:

$$\begin{aligned}\mathbb{E}[\ln x] &= -\ln\beta + \Psi(\alpha)\\ \mathbb{E}[x\ln x] &= \frac{\alpha}{\beta}\left(\Psi(\alpha+1) - \ln\beta\right)\\ \mathbb{E}[\ln^2 x] &= \left(\ln\beta - \Psi(\alpha)\right)^2 + \Psi_1(\alpha)\end{aligned}$$

Replacing the moments for the following equations: $\mathbb{E}[G_{11}]$, $\mathbb{E}[G_{22}]$ and $\mathbb{E}[G_{12}]$ will finally give the Fisher-Rao matrix.

Von Mises Fisher distribution, for $\mathbb{S}^2$:

We have $p(\boldsymbol{x}|\mu,\kappa) = C_3(\kappa)\exp\left(\kappa\mu^\top\boldsymbol{x}\right)$, with $C_3(\kappa) = \kappa(4\pi\sinh\kappa)^{-1}$. Here, μ is a 3-dimensional vector with $\|\mu\| = 1$.

$$\begin{aligned}\ln p(\boldsymbol{x}|\mu,\kappa) &= \ln\kappa - \ln 4\pi - \ln\sinh(\kappa) + \kappa\mu^\top\boldsymbol{x}\\ \nabla_\mu \ln p &= \kappa\boldsymbol{x}\\ \frac{\partial \ln p}{\partial\kappa} &= \kappa^{-1} - \coth(\kappa) + \mu^\top\boldsymbol{x}.\end{aligned}$$

Here, the Fisher-Rao matrix $\mathbf{I}_{\mathcal{S}}$ will be composed of block matrices, such that: $\mathbf{I}_{\mathcal{S}} = \mathbb{E}[G]$, with G_{11} a 3×3-matrix, G_{22} a scalar, and $G_{12} = G_{21}^\top$ a 3-dimensional vector.

$$\begin{aligned}G_{11} &= \nabla_\mu \ln p \nabla_\mu \ln p^\top = \kappa^2\boldsymbol{x}\boldsymbol{x}^\top\\ G_{22} &= \left(\frac{\partial \ln p}{\partial\kappa}\right)^2 = K(\kappa)^2 + 2K(\kappa)\mu^\top\boldsymbol{x} + (\mu^\top\boldsymbol{x})^2\\ G_{12} = G_{21}^\top &= \frac{\partial \ln p}{\partial\kappa}\cdot\nabla_\mu \ln p = \left(K(\kappa) + \mu^\top\boldsymbol{x}\right)\kappa\boldsymbol{x},\end{aligned}$$

with $K(\kappa) = \coth(\kappa) - \frac{1}{\kappa}$.

We know from Hillen et al. [2016] that the mean and variance of the Von Mises Fisher distribution in the 3-dimensional case is: $\mathbb{E}[\boldsymbol{x}] = K(\kappa)\mu$ and $\mathrm{Var}[\boldsymbol{x}] = \frac{1}{\kappa}K(\kappa)\mathbb{1} + (1 - \frac{\coth(\kappa)}{\kappa} + \frac{2}{\kappa^2} - \coth^2(\kappa))\mu\mu^\top$. We can then deduce the following meaningful moments:

$$\begin{aligned}\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top] &= \mathrm{Var}[\boldsymbol{x}] + \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{x}]^\top = \left(1 - \frac{3}{\kappa}K(\kappa)\right)\mu\mu^\top + \frac{1}{\kappa}K(\kappa)\mathbb{1},\\ \mathbb{E}[\mu^\top\boldsymbol{x}] &= \mu^\top\mathbb{E}[\boldsymbol{x}] = K(\kappa)\mu^\top\mu = K(\kappa)\\ \mathbb{E}[(\mu^\top\boldsymbol{x})^2] &= \mu^\top\mathrm{Var}[\boldsymbol{x}]\mu + \mathbb{E}[\mu^\top\boldsymbol{x}]^2 = 1 - \frac{2}{\kappa}K(\kappa),\\ \mathbb{E}[\mu^\top\boldsymbol{x}\boldsymbol{x}] &= \mathbb{E}[\mu\boldsymbol{x}\boldsymbol{x}^\top] = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top]\mu = \left(1 - \frac{3}{\kappa}K(\kappa)\right)\mu + \frac{1}{\kappa}K(\kappa)\mu.\end{aligned}$$

Replacing those moments in the following expressions: $\mathbb{E}[G_{11}]$, $\mathbb{E}[G_{22}]$, $\mathbb{E}[G_{12}]$ directly gives the Fisher-Rao metric. □

B.2 Curve energy approximation for categorical data

In this section we present the details of the example in Section 3.3.3. In particular, we the steps to derive an approximation to the energy of a latent curve in closed form, which is suitable for applying automatic differentiation. This is particularly useful for our setting, since it allows us to consider our framework as a Black Box Random Geometry processing toolbox.

Let a random variable $\boldsymbol{x} \in \mathbb{R}^D$ that follows a generalized Bernoulli likelihood $p(\boldsymbol{x}|\eta)$, so the vector $\boldsymbol{x} \in \mathbb{R}^D$ is of the form $\boldsymbol{x} = (0, \cdots, 1, \cdots, 0)$ with $\sum_i x_i = 1$. The parameters $\eta \in \mathbb{R}^D$ are given as $\eta = h(\boldsymbol{z})$, with $\eta_i \geq 0 \ \forall i$ and $\sum_i \eta_i = 1$ so we know that the parameters lie on the unit simplex. Actually, they represent the probability the corresponding dimension to be 1 on a random draw. Also, the $p(\boldsymbol{x}|\boldsymbol{z}) = \eta_1^{[x_1]} \cdots \eta_D^{[x_D]}$, where $[x_i] = 1$ if $x_i = 1$ else $[x_i] = 0$ which can be seen as an indicator function. The $\log p(\boldsymbol{x}|\eta) = \sum_i [x_i] \log(\eta_i)$ and $\nabla_\eta \log p(\boldsymbol{x}|\eta) = \left(\frac{[x_1]}{\eta_1}, \ldots, \frac{[x_D]}{\eta_D} \right)$. Due to the outer product we have to compute the following expectations

$$\mathbb{E}_{\boldsymbol{x}} \left[\frac{[x_i]}{\eta_i} \frac{[x_j]}{\eta_j} \right] = 0, \quad \text{if} \quad i \neq j, \tag{B.3}$$

$$\mathbb{E}_{\boldsymbol{x}} \left[\left(\frac{[x_i]}{\eta_i} \right)^2 \right] = \frac{1}{\eta_i}, \quad \text{if} \quad i = j, \tag{B.4}$$

because the $[x_i]$ and $[x_j]$ cannot be 1 on the same time, while the $\mathbb{E}_{\boldsymbol{x}}[[x_i]^2] = \eta_i$ as it shows the number of times $x_i = 1$. So the Fisher-Rao metric of $\mathcal{H}$ is equal to $I_{\mathcal{H}}(\eta) = \operatorname{diag}(1/\eta_1, \ldots, 1/\eta_D)$. Note that the shortest paths between two distributions must be on the unit simplex in $\mathcal{H}$, while on the same time respecting the geometry of the Fisher-Rao metric.

We can easily parametrize the unit simplex by $[\eta_1, \ldots, \eta_{D-1}, \widetilde{\eta}_D]$ with

$$\widetilde{\eta}_D(\eta_1, \ldots, \eta_{D-1}) = 1 - \sum_{i=1}^{D-1} \eta_i. \tag{B.5}$$

This allows to pullback the Fisher-Rao metric in the latent space $[\eta_1, \ldots, \eta_{D-1}]$ as we have described in this paper. Intuitively, the $\boldsymbol{z} = [\eta_1, \ldots, \eta_{D-1}]$ and the function h is the parametrization of the simplex. Hence, we are able to compute the shortest path using the induced metric.

However, there is a simpler way to compute this path. We know that the element-wise square root of the parameters η gives a point on the positive orthonant of the unit sphere as $y_i = \sqrt{\eta_i} \ \Rightarrow \ \sum_i y_i^2 = \sum_i \sqrt{\eta_i}^2 = 1$. We also know that the shortest path on a sphere is the great-circle. Therefore, the distance between two distributions parametrized by η and η' on the unit simplex in $\mathcal{H}$, can be equivalently measured

using the great-circle distance between their square roots as

$$\text{dist}(\eta, \eta') = \arccos \sqrt{\eta}^\top \sqrt{\eta'}. \tag{B.6}$$

In this way, we can approximate the energy of a curve $c(t)$ in the latent space as follows

$$\begin{aligned}
\text{Energy}[c] &\approx \sum_{n=1}^{N-1} \text{dist}^2(h(c(n/N)), h(c(n+1/N))) && \text{(B.7)}\\
&= \sum_{n=1}^{N-1} \arccos^2 \sqrt{h(c(n/N))}^\top \sqrt{h(c(n+1/N))} \\
&= \sum_{n=1}^{N-1} \left(2 - 2\sqrt{h(c(n/N))}^\top \sqrt{h(c(n+1/N))}\right), && \text{(B.8)}
\end{aligned}$$

where we used at the last step the small angle approximation $\cos\theta \approx 1 - \frac{\theta^2}{2} \Leftrightarrow \theta^2 \approx 2 - 2\cos\theta$. Note that this formulation is suitable for our proposed method to compute shortest paths (see Section 3.3.2).

The derivation above represents the conceptual strategy, while in general we proposed to use the KL divergence approximation result equation 3.8 in place of the great-circle distance. Intuitively, when the KL divergence has an analytic solution, we can derive an analogous energy approximation. Even if the solution of the KL is intractable, we can still use our approach as long as we can estimate the KL using Monte Carlo and propagate the gradient through the samples using a re-parametrization scheme or a score function estimator.

B.3 Information geometry in generative modeling

In this section we present the additional technical information related to the pullback Fisher-Rao metric in the latent space of a VAE.

B.3.1 Details for the pullback metric in the latent space

We call h the non linear function, typically parametrized as deep neural networks, that maps the variables from the latent space $\mathcal{Z}$ to the parameter space $\mathcal{H}$, such that: $h(\boldsymbol{z}) = \eta$, with $\boldsymbol{z} \in \mathcal{Z}$ and $\eta \in \mathcal{H}$. Furthermore, the data $\boldsymbol{x} \in \mathcal{X}$ is reconstructed such that it follows a specific distribution: $\boldsymbol{x} \sim p(\boldsymbol{x}|\eta)$, with $p(\boldsymbol{x}|\eta)$ being for instance a Bernoulli or Gaussian distribution. The parameter space $\mathcal{H}$ is a statistical manifold equipped with Fisher-Rao metric: $I_{\mathcal{H}}(\eta) \triangleq \int p(\boldsymbol{x}|\eta) \left[\nabla_\eta \log p(\boldsymbol{x}|\eta) \nabla_\eta \log p(\boldsymbol{x}|\eta)^\top\right] d\boldsymbol{x}$. We denote by J_h the Jacobian of h.

Proposition 7. *The latent space $\mathcal{Z}$ is equipped with the Riemannian pullback metric tensor:*

$$G(\boldsymbol{z}) \triangleq J_h(\boldsymbol{z})^\top I_{\mathcal{H}}(h(\boldsymbol{z}))J_h(\boldsymbol{z}).$$

Proof. The parameter space is a statistical manifold equipped with the Fisher-Rao metric $I_{\mathcal{H}}(\eta)$, thus the scalar product at η between two vectors $d\eta_1, d\eta_2 \in \mathcal{H}$ is: $\langle d\eta_1, d\eta_2 \rangle_{I_{\mathcal{H}}(\eta)} = d\eta_1^\top I_{\mathcal{H}}(\eta) d\eta_2$. For two vectors $d\boldsymbol{z}_1, d\boldsymbol{z}_2 \in \mathcal{Z}$, we have at $\eta = f(\boldsymbol{z})$ that: $\langle d\eta_1, d\eta_2 \rangle_{I_{\mathcal{H}}(\eta)} = \langle J_h(\boldsymbol{z}) d\boldsymbol{z}_1, J_h(\boldsymbol{z}) d\boldsymbol{z}_2 \rangle_{I_{\mathcal{H}}(\eta)} = d\boldsymbol{z}_1^\top (J_h(\boldsymbol{z})^\top I_{\mathcal{H}}(h(\boldsymbol{z}))J_h(\boldsymbol{z})) d\boldsymbol{z}_2$.

$I_{\mathcal{H}}(h(\boldsymbol{z}))$ is a Riemannian metric tensor by definition, and it is then positive definite. Furthermore, $h : \mathcal{Z} \to \mathcal{H}$ is a smooth immersion, and so $J_h(\boldsymbol{z})$ is full-rank. It follows that $J_h(\boldsymbol{z})^\top I_{\mathcal{H}}(h(\boldsymbol{z}))J_h(\boldsymbol{z})$ is positive definite. Hence $G(\boldsymbol{z})$ is a Riemannian metric tensor. □

Proposition 8. *Our pullback metric $G(\boldsymbol{z})$ is actually equal to the Fisher-Rao metric obtained over the parameter space $\mathcal{Z}$:*

$$G(\boldsymbol{z}) = I_{\mathcal{Z}}(\boldsymbol{z}) \triangleq \int p(\boldsymbol{x}|\boldsymbol{z}) \left[\nabla_{\boldsymbol{z}} \log p(\boldsymbol{x}|\boldsymbol{z}) \nabla_{\boldsymbol{z}} \log p(\boldsymbol{x}|\boldsymbol{z})^\top \right] d\boldsymbol{x}$$

Proof. We will show that $I_{\mathcal{Z}}(\boldsymbol{z}) = J_f(\boldsymbol{z})^\top I_{\mathcal{H}}(\eta) J_f(\boldsymbol{z})$. Let's consider the definition of the Fisher-Rao metric in $\mathcal{Z}$:

$$\begin{aligned}
I_{\mathcal{Z}}(\boldsymbol{z}) &= \int \nabla_{\boldsymbol{z}} \log p(\boldsymbol{x} \mid \boldsymbol{z}) \cdot \nabla_{\boldsymbol{z}} \log p(\boldsymbol{x} \mid \boldsymbol{z})^\top p(\boldsymbol{x} \mid \boldsymbol{z}) d\boldsymbol{x} && \text{(B.9)} \\
&= \int J_f(\boldsymbol{z})^\top \nabla_\eta \log p(\boldsymbol{x}|\eta) \nabla_\eta \log p(\boldsymbol{x}|\eta)^\top J_f(\boldsymbol{z}) p(\boldsymbol{x}|\eta) d\boldsymbol{x} && \text{(B.10)} \\
&= J_f(\boldsymbol{z})^\top \left[\int_{\mathcal{X}} \nabla_\eta \log p(\boldsymbol{x}|\eta) \nabla_\eta \log p(\boldsymbol{x}|\eta)^\top p(\boldsymbol{x}|\eta) d\boldsymbol{x} \right] J_f(\boldsymbol{z}) && \text{(B.11)} \\
&= J_f(\boldsymbol{z})^\top I_{\mathcal{H}}(f(\boldsymbol{z})) J_f(\boldsymbol{z}) = G(\boldsymbol{z})
\end{aligned}$$

where we use the fact that $\eta = f(\boldsymbol{z})$ so the $\nabla_{\boldsymbol{z}} \log p(\boldsymbol{x}|f(\boldsymbol{z})) = J_f(\boldsymbol{z})^\top \cdot \nabla_\eta \log p(\boldsymbol{x}|\eta)$

The same argument can be proved as follows:

$$\begin{aligned}
&\langle d\eta, I_{\mathcal{H}}(\eta) d\eta \rangle = \langle J_f(\boldsymbol{z}) d\boldsymbol{z}, I_{\mathcal{H}}(f(\boldsymbol{z})) J_f(\boldsymbol{z}) d\boldsymbol{z} \rangle && \text{(B.12)} \\
&= \langle J_f(\boldsymbol{z}) d\boldsymbol{z}, \int \nabla_\eta \log p(\boldsymbol{x}|\eta) \nabla_\eta \log p(\boldsymbol{x}|\eta)^\top p(\boldsymbol{x}|\eta) d\boldsymbol{x} \; J_f(\boldsymbol{z}) d\boldsymbol{z} \rangle && \text{(B.13)} \\
&= \langle d\boldsymbol{z}, \int J_f(\boldsymbol{z})^\top \cdot \nabla_\eta \log p(\boldsymbol{x}|\eta) \; \nabla_\eta \log p(\boldsymbol{x}|\eta)^\top \cdot J_f(\boldsymbol{z}) p(\boldsymbol{x}|\eta) d\boldsymbol{x} \; d\boldsymbol{z} \rangle && \text{(B.14)} \\
&= \langle d\boldsymbol{z}, \int \nabla_{\boldsymbol{z}} \log p(\boldsymbol{x}|\boldsymbol{z}) \nabla_{\boldsymbol{z}} \log p(\boldsymbol{x}|\boldsymbol{z})^\top p(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{x} \; d\boldsymbol{z} \rangle = \langle d\boldsymbol{z}, I_{\mathcal{Z}}(\boldsymbol{z}) d\boldsymbol{z} \rangle && \text{(B.15)}
\end{aligned}$$

□

In section B.1.1, we have seen how to derive a close-form expression of the Fisher-Rao metric for a one-dimensional observation x that follows a specific distribution. In practice, $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^D$ is a multi-dimensional variable where each dimension represents, for instance, a pixel when working with images or a feature when working with tabular data. Each feature, x_i with $i = 1 \cdots D$, is obtained for a specific set of parameters $\{\eta_i\}$. We assume that the features follow the same distribution $\mathcal{D}$, such that: $x_i \sim p(x_i|\eta_i)$, and $p(\boldsymbol{x}|\eta) = \prod_{i=1}^{D} p(x_i|\eta_i)$.

Proposition 9. *If the features follow the same distribution $\mathcal{D}$, such that: $x_i \sim p(x_i|\eta_i)$ and $p(\boldsymbol{x}|\eta) = \prod_{i=1}^{D} p(x_i|\eta_i)$, then the Fisher-Rao metric $\mathbf{I}_{\mathcal{H}}(\eta)$ is a block matrix where the diagonal terms are the Fisher-Rao matrices $\mathbf{I}_{\mathcal{H},i}$ obtained for each data feature x_i:*

$$\mathbf{I}_{\mathcal{H}}(\eta) = \begin{pmatrix} I_{\mathcal{H},1} & 0 & \dots & 0 \\ 0 & I_{\mathcal{H},2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I_{\mathcal{H},D} \end{pmatrix}$$

Proof. We have $x_i \sim p(x_i|\eta_i)$ and $\mathbf{I}_{\mathcal{H},i} = \int p(x_i|\eta_i) \left[\nabla_{\eta_i} \log p(x_i|\eta_i) \nabla_{\eta_i} \log p(x_i|\eta_i)^\top\right] dx_i$. Also, we assumed: $p(\boldsymbol{x}|\eta) = \prod_{i=1}^{D} p(x_i|\eta_i)$. We then have: $\log p(\boldsymbol{x}|\eta) = \sum_{i=1}^{D} \log p(x_i|\eta_i)$, and the Fisher score:

$$\nabla_\eta \log p(\boldsymbol{x}|\eta) = \nabla_\eta \sum_{i=1}^{D} \log p(x_i|\eta_i) = \left[\nabla\eta_1 \ln p(x_1|\eta_1), \dots, \nabla\eta_D \ln p(x_1|\eta_D)\right]^\top \tag{B.16}$$

The matrix $\mathbf{I}_{\mathcal{H}}(\eta)$ is thus a $D \times D$ block matrix, where the (i, j)-block element is:

$$I_{ij} = \int p(x_i|\eta_i) \left[\nabla_{\eta_i} \log p(x_i|\eta_i) \nabla_{\eta_i} \log p(x_j|\eta_j)^\top\right] dx_i.$$

Let's note that:

$$\int p(x_i|\eta_i) \nabla_{\eta_i} \log p(x_i|\eta_i) dx_i = \int p(x_i|\eta_i) \frac{\nabla_{\eta_i} p(x_i|\eta_i)}{p(x_i|\eta_i)} dx_i = \nabla_{\eta_i} \int p(x_i|\eta_i) dx_i = 0.$$

When $i = j$, we have $\mathbf{I}_{ii} = \mathbf{I}_{\mathcal{H},i}$, with $\mathbf{I}_{\mathcal{H},i}$ being the Fisher-Rao metric obtained for: $x_i \sim p(x_i|\eta_i)$.
When $i \neq j$, we have: $\mathbf{I}_{ij} = \nabla \log p(x_j|\eta_j)^\top \int p(x_i|\eta_i) \nabla_{\eta_i} \log p(x_i|\eta_i) dx_i = 0$. □

Then, for example, if we are dealing with binary images, and make the assumption that each pixel x_i follows a Bernoulli distribution: $p(x_i|\eta_i) = \eta^{x_i}(1 - \eta_i)^{1-x_i}$, then according to Section B.1.1 and Proposition 9, the Fisher-Rao matrix that endows the

parameter space $\mathcal{H}$ is:

$$I_{\mathcal{H}}(\eta) = \begin{pmatrix} \frac{1}{\eta_1(1-\eta_1)} & 0 & \dots & 0 \\ 0 & \frac{1}{\eta_2(1-\eta_2)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\eta_D(1-\eta_D)} \end{pmatrix}.$$

We have seen that in theory, we can obtain a close form expression for the pullback metric, if the probability distribution is known. In practice, we can directly infer the metric using the approximation of the KL-divergence.

Proposition 10. *We define perturbations vectors as: $\delta e_i = \varepsilon \cdot \mathbf{e_i}$, with $\varepsilon \in \mathbb{R}_+$ a small infinitesimal quantity, and ($\mathbf{e_i}$) a canonical basis vector in $\mathbb{R}^d$. For clarity, we rename $D_{\mathrm{KL}}(p(\boldsymbol{x}|\boldsymbol{z})||p(\boldsymbol{x}|\boldsymbol{z}+\delta\boldsymbol{z})) = \mathrm{KL}_{\boldsymbol{z}}(\delta\boldsymbol{z})$ and we note G_{ij} the components of $G(\boldsymbol{z})$. We can then approximate by a system of equations the diagonal and non-diagonal elements of the metric:*

$$G_{ii} \approx 2\ \mathrm{KL}_{\boldsymbol{z}}(\delta\mathbf{e_i})/\varepsilon^2$$
$$G_{ij} = G_{ji} \approx \left(\mathrm{KL}_{\boldsymbol{z}}(\delta\mathbf{e_i} + \delta\mathbf{e_j}) - \mathrm{KL}_{\boldsymbol{z}}(\delta\mathbf{e_i}) - \mathrm{KL}_{\boldsymbol{z}}(\delta\mathbf{e_j})\right)/\varepsilon^2.$$

Proof. From Proposition 5, we know that:

$$\mathrm{KL}_{\boldsymbol{z}}(\delta\boldsymbol{z}) = \frac{1}{2}\delta\boldsymbol{z}^\top G(\boldsymbol{z})\delta\boldsymbol{z} + o(\delta\boldsymbol{z}^2).$$

Let's take $\delta e_i = \varepsilon \cdot \mathbf{e_i}$. On one hand, we have: $\delta e_i^\top G(\boldsymbol{z})\delta e_i = \varepsilon^2 G_{ii}$. On the second hand, we also have: $\delta e_i^\top G(\boldsymbol{z})\delta e_i \approx 2\mathrm{KL}_{\boldsymbol{z}}(\delta\mathbf{e_i})$, which gives us the equation to infer the diagonal elements of the metric.

Now, let's take $\delta e_i + \delta e_j = \varepsilon \cdot (\mathbf{e_i} + \mathbf{e_j})$. Then, we have: $(\delta e_i + \delta e_j)^\top G(\boldsymbol{z})(\delta e_i + \delta e_j) = \varepsilon^2(G_{ii} + G_{jj} + G_{ij} + G_{ji})$. We also know that $G_{ji} = G_{ij}$. Again, we also have: $(\delta e_i + \delta e_j)^\top G(\boldsymbol{z})(\delta e_i + \delta e_j) \approx 2\mathrm{KL}_{\boldsymbol{z}}(\delta\mathbf{e_i} + \delta\mathbf{e_j})$.

We can replace the terms G_{ii} and G_{jj} in the equation obtained above with the KL-divergence for the diagonal terms. Which finally gives us:

$$G_{ij} = G_{ji} \approx \left(\mathrm{KL}_{\boldsymbol{z}}(\delta\mathbf{e_i} + \delta\mathbf{e_j}) - \mathrm{KL}_{\boldsymbol{z}}(\delta\mathbf{e_i}) - \mathrm{KL}_{\boldsymbol{z}}(\delta\mathbf{e_j})\right)/\varepsilon^2. \tag{B.17}$$

□

B.3.2 Uncertainty quantification and regularization

As discussed in the main text, we carefully design our mappings from latent space to parameter space such that they model the training codes according to the learned

decoders, and extrapolate to uncertainty outside the support of the data. This, we refer to as **uncertainty regularization**. In this section we explain it in detail. The core idea of this uncertainty regularization is imposing a "slider" that forces the distribution $p(\boldsymbol{x}|\boldsymbol{z})$ to change when $\boldsymbol{z}$ is far from the training latent codes. For this, we use a combination of KMeans and the sigmoid activation function.

We start by encoding our training data, arriving at a set of latent codes $\{\boldsymbol{z}_n\}_{n=1}^N \subseteq \mathcal{Z}$. We then train KMeans($k$) on these latent codes (where k is a hyperparameter that we tweak manually), arriving at k cluster centers $\{c_j\}_{j=1}^k$. These cluster centers serve as a proxy for "closeness" to the data: we know that a latent code $\boldsymbol{z} \in \mathcal{Z}$ is near the support if $D(\boldsymbol{z}) := \min_j \left\{\|\boldsymbol{z} - c_j\|^2\right\}$ is close to 0.

The next step in our regularization process is to reweight our decoded distributions such that we decode to high uncertainty when $D(\boldsymbol{z})$ is large, and we decode to our learned distributions when $D(\boldsymbol{z}) \approx 0$. This mapping from $[0, \infty) \to (0, 1)$ can be constructed using a modified sigmoid function Detlefsen et al. [2019b, 2020], consider indeed

$$\tilde{\sigma}_\beta(d) = \text{Sigmoid}\left(\frac{d - c \cdot \text{Softplus}(\beta)}{\text{Softplus}(\beta)}\right), \tag{B.18}$$

where $\beta \in \mathbb{R}$ is another hyperparameter that we manually tweak, and $c \approx 7$.

With this translated sigmoid, we have that $\tilde{\sigma}_\beta(D(\boldsymbol{z}))$ is close to 0 when $\boldsymbol{z}$ is close to the support of the data (i.e. close to the cluster centers), and it converges to 1 when $D(\boldsymbol{z}) \to \infty$. $\tilde{\sigma}_\beta(D(\boldsymbol{z}))$ serves, then, as a slider that indicates closeness to the training codes. This reweighting takes the following form:

$$\text{reweight}(\boldsymbol{z}) = (1 - \tilde{\sigma}_\beta(D(\boldsymbol{z})))h(\boldsymbol{z}) + \tilde{\sigma}_\beta(D(\boldsymbol{z}))\,\text{extrapolate}(\boldsymbol{z}), \tag{B.19}$$

where $h(\boldsymbol{z}) = \eta \in \mathcal{H}$ represents our learned networks in parameter space, and extrapolate($\boldsymbol{z}$) returns the parameters of the distribution that maximize uncertainty (e.g. $\sigma \to \infty$ in the case of an isotropic Gaussian, $p \to 1/2$ in the case of a Bernoulli, and $\kappa \to 0$ in the case of the von Mises-Fisher).

For the particular case of the experiment in which we pull back the Fisher-Rao metric from the parameter space of several distributions (see 3.4.2), Table B.2 provides the exact extrapolation mechanisms and implementations of $h(\boldsymbol{z})$.

B.4 Details for our implementation and experiments

In this section we present the technical details that we used in our implementation and experiments. We are currently implementing an open-source version of our code here.

Distribution	$h\colon \mathcal{Z}_{\text{toy}} \to \mathcal{H}$	Extrapolation mechanism
Normal	$\mu(\boldsymbol{z}) = 10 \cdot f_3(\boldsymbol{z}), \quad \sigma(\boldsymbol{z}) = 10 \cdot \text{Softplus}(f_3(\boldsymbol{z}))$	$\sigma(\boldsymbol{z}) \to \infty$
Bernoulli	$p(\boldsymbol{z}) = \text{Sigmoid}(f_{15}(\boldsymbol{z}))$	$p(\boldsymbol{z}) = 1/2$
Beta	$\alpha(\boldsymbol{z}) = 10 \cdot \text{Softplus}(f_3(\boldsymbol{z})), \quad \beta(\boldsymbol{z}) = 10 \cdot \text{Softplus}(f_3(\boldsymbol{z}))$	$(\alpha(\boldsymbol{z}), \beta(\boldsymbol{z})) = (1, 1)$
Dirichlet	$\alpha(\boldsymbol{z}) = \text{Softplus}(f_3(\boldsymbol{z}))$	$\alpha(\boldsymbol{z}) = 1$
Exponential	$\lambda(\boldsymbol{z}) = \text{Softplus}(f_3(\boldsymbol{z}))$	$\lambda(\boldsymbol{z}) \to 0$

Table B.2. This table shows the implementations of the decode and extrapolate functions in Eq. (B.19) for all the distributions studied in our second experiment (see Sec. 3.4.2). Here we represent a randomly initialized neural network with f_i, where i represents the size of the co-domain. For example, in the case of the Dirichlet distribution, we use a randomly initialized neural network to compute the parameters α of the distribution and, since these have to be positive, we pass the output of this network through a Softplus activation; moreover, since the Dirichlet distribution is approximately uniform when all its parameters equal 1, our extrapolation mechanism consists of replacing the output of the network with a constant vector of ones.

B.4.1 What we mean when we say black-box random geometry

Before we dive into the specific details of our experiments, it is worth noting that they were all made using the same *interface*. This is precisely what we mean when we say that our results open the doors for black-box random geometry: We can define a `curve_energy` method that is agnostic to the distribution our models decode to.

To hammer this point home, consider the following interface, written in Python:

```python
class StatisticalManifold:
    def __init__(self, model: torch.nn.Module):
        # A model with regularized uncertainty (see Uncertainty Quantification)
        self.model = model
        assert "decode" in dir(model)

    def curve_energy(self, curve: CubicSpline) -> torch.Tensor:
        # An energy function that can be minimized using autodifferentiation.

        dt = (curve[1] - curve[0])
        dist1 = self.model.decode(curve[:-1])
        dist2 = self.model.decode(curve[1:])
        kl = kl_divergence(dist1, dist2)
        energy = kl.sum() * (2 * dt ** -1)

        return energy
```

Notice that the user need only provide a `model` that implements a `decode` function which is expected to return a distribution with proper uncertainty estimates (as de-

scribed in Sec. B.3.2). Line 14 is a direct implementation of our derived expression for the energy (see Prop. 6). Most distributions of interest are available in the Torch submodule `torch.distributions`, and similar implementations could be done for other frameworks.

B.4.2 Shortest path approximation with cubic splines

As we described in the main paper, we use an approximate solution for the shortest paths based on cubic splines. Let a cubic spline

$$c_{\backslash\psi}(t) = [1, t, t^2, t^3]^\top [\backslash\psi_0, \backslash\psi_1, \backslash\psi_2, \backslash\psi_3] \tag{B.20}$$

with parameters $\backslash\psi_i \in \mathbb{R}^{d\times 1}$. Also, in our implementation the actual curve is a piecewise cubic spline and we optimize the K control points c_k as well. We optimize the parameters using the approximation of the curve energy $\{\backslash\psi_k^*, c_k^*\}_{k=1}^K = \arg\min_{\backslash\psi} \text{Energy}[c_{\backslash\psi}]$. In general, we can use Prop. 6 as long as we can propagate the gradient through the KL or as in equation B.7 if an explicit closed form solution exists. In this case, we are able to use automatic differentiation for the optimization of the parameters (as discussed in Sec. B.4.1).

In practical terms, we compute these shortest paths by creating a uniform grid in latent space and computing, only once, the curve energy for the edges of this grid. After this expensive computation (which only needs to be performed once) we can use shortest-paths algorithms in graphs to create a suitable initialization of the geodesic. We fit a cubic spline to this initialization and then optimize its parameters further.

B.4.3 Models used

In this section we describe, in detail, the models that we used for our experiments (see Sec.3.4). All the networks that we used are Multi-Layer Perceptrons implemented in PyTorch.

First, Table B.3 shows the Variational Autoencoder implemented for the experiment described in Sec. 3.4.1. In the computations *without* uncertainty regularization, we used a simpler model for the uncertainty quantification (namely, a single Linear layer, followed by a Softplus activation). For our second experiment involving a toy latent space, we also provide the implementation of the respective MLPs in Table B.4. Finally, Table B.5 and B.6 respectively represents the VAE trained for the experiments related to motion capture (Sec. 3.4.3) and movie rating (Sec. 3.4.6). For the motion capture experiments, we are training a VAE that decodes to a von Mises Fisher distribution, and for the movie rating experiments, we decode to a Bernoulli distribution.

Pulling back the Euclidean vs. Fisher-Rao (Sec. 3.4.1)

Module	MLP
	Encoder
μ	Linear(728, 2)
	Decoder
μ	Linear(2, 728)
σ_{UR}	RBF(), PosLinear(500, 1), Reciprocal(), PosLinear(1, 728)
$\sigma_{\text{no UR}}$	Linear(2, 728), Softplus()
Optimizer	Adam ($\alpha = 1 \times 10^{-5}$)
Batch size	32

Table B.3. This table shows the Variational Autoencoder used in our first experiment (see Sec. 3.4.1). The network for approximating the standard deviation σ leverages ideas from Arvanitidis et al. [2018], in which an RBF network is trained on latent codes using centers positioned through KMeans. The operation PosLinear(a, b) represents the usual Linear transformation with a inputs and b outputs, but considering only positive weights. To compare between having and not having uncertainty regularization, we use two different approximations of the standard deviation in the decoder: σ_{UR} when performing meaningful uncertainty quantification, and $\sigma_{\text{no UR}}$ otherwise.

Toy latent spaces (Sec. 3.4.2)

Distribution	Module	MLP	Random seed	β in $\tilde{\sigma}_\beta$
Normal	μ σ	Linear(2,3) Linear(2,3), Softplus()	1	-2.5
Bernoulli	p	Linear(2,15), Sigmoid()	1	-3.5
Beta	α β	Linear(2,3), Softplus() Linear(2,3), Softplus()	1	-4.0
Dirichlet	α	Linear(2,3), Softplus()	17	-4.0
Exponential	λ	Linear(2,3), Softplus()	17	-4.0

Table B.4. This table describes the neural networks used for the experiment presented in Sec. 3.4.2. Following the notation of PyTorch, Linear(a, b) represents an MLP layer with a input nodes and b output nodes. In each of these networks, we implement the reweighting operation described in Sec. B.3.2, and we describe the β hyperparameter present in the modified sigmoid function (Eq. (B.18)). This networks were not trained in any way, and they were initialized using the provided seed.

Decoding to a von Mises-Fisher Distribution (Sec 3.4.3, 3.4.4, 3.4.5)

Module	MLP
	Encoder (Normal dist.)
μ	Linear(3×26, 90), Linear(90, 2)
σ	Linear(3×26, 90), Linear(90, 2), Softplus()
	Decoder (vMF dist.)
μ	Linear(2, 90), Linear(90, 3×26), Linear(3×26, 3×26)
κ	Linear(2, 90), Linear(90, 3×26), Linear(3×26, 26), Softplus()
Optimizer	Adam ($\alpha = 1 \times 10^{-3}$)
Batch size	16
β in $\tilde{\sigma}_\beta$	-5.5
KL annealing	0.01
Extrapolation	$\kappa \to 0.1$

Table B.5. This table shows the Variational Autoencoder used in our last two experiments (see Sec. 3.4.3, 3.4.4). Our motion capture data tracked 26 different bones, and thus we decode to a product of 26 different von Mises-Fisher distributions.

Decoding to a Bernoulli Distribution (Sec 3.4.6)

Module	MLP
	Encoder (Normal dist.)
μ	Linear(60, 16), Tanh(), Linear(16, 16), Tanh(), Linear(16, 2)
σ	Linear(60, 16), Tanh(), Linear(16, 16), Tanh(), Linear(16, 2), Softplus()
	Decoder (Bernoulli dist.)
p	Linear(2, 16), Tanh(), Linear(16, 16), Tanh(), Linear(16, 60), Sigmoid()
Optimizer	Adam ($\alpha = 1 \times 10^{-3}$, $\omega = 1 \times 10^{-7}$)
Batch size	256
β in $\tilde{\sigma}_\beta$	-3.0
KL annealing	0.01
Extrapolation	$p \to 1/2$

Table B.6. This table shows the Variational Autoencoder used in the movie rating experiement (see Sec. 3.4.6). The MovieLens 25M dataset has been preprocessed such that it is composed of 10000 users rating if they have seen some of 60 selected movies. We only select users that have seen more than two movies and less than 30 movies, to avoid outliers and aim for a more realistic scenario. We used the same extrapolation mechanism described in the toy experiments for the Bernoulli: having the probits be 1/2 (see Sec. B.3.2).

All of these VAEs were trained by maximizing the Evidence Lower Bound with different values for KL annealing which can be read from the different tables. For example, Table B.5 shows that the KL annealing constant was chosen to be 0.01.

B.4.4 Metric approximation and KL by sampling

When visualising our latent space as a statistical manifold, we can obtain a direct approximation of the metric using the KL-divergence between two close distributions (Proposition 4). We will show here, in simple cases, how our metric approximation compares to close-form expressions.

In the following experiment, our statistical manifold is the parameter space of known distributions (Beta and Normal). Their Fisher-Rao matrices are well-known (Sec. B.1.1), and we approximate them by computing the KL-divergence of sampled distributions. We call G_t the theoretical metric and G_a the approximated metric, and we note $\varepsilon_r = \frac{\|G_t - G_a\|}{\|G_t\|}$ the relative error between the theoretical and approximated matrices. Here, $\|\cdot\|$ denotes the Frobenius norm. For the Normal distribution, we empirically obtain: $\varepsilon_r = 5.32 \cdot 10^{-4} \pm 9.63 \cdot 10^{-4}$, and for the Beta distribution, we have: $\varepsilon_r = 1.73 \cdot 10^{-5} \pm 1.17 \cdot 10^{-5}$.

B.4.5 Computational complexity

Proposition 5 shows the system of equations required to approximate the pullback metric in the latent space. Each KL operation requires 2 forward passes from the decoder to compute, so first we establish the lower bound on the time complexity of the decoder forward pass. Ignoring all activation function related operations, for an MLP with H hidden layers, N-dimensional network output, K-dimensional hidden layer output and single M-dimensional vector input, this lower bound is:

$$\Omega\left(MK_1 + K_H N + \sum_{i=1}^{H-1} M_i M_{i+1}\right) \tag{B.21}$$

For each diagonal element G_{ii} of the metric tensor we need to compute a single KL divergence, which will require two forward passes through the decoder network giving us a (lower bounded) time complexity of $\Omega\left[2\left(MK_1 + K_H N + \sum_{i=1}^{H-1} M_i M_{i+1}\right)\right]$ for each element. For the off-diagonal elements we will need to compute the KL three times which corresponds to six forward passes through the decoder network. which yields a (lower bounded) time complexity $\Omega\left[6\left(MK_1 + K_H N + \sum_{i=1}^{H-1} M_i M_{i+1}\right)\right]$ per element.

B.4.6 Information for the movie preferences experiment

For this experiment we used the MovieLens 25M dataset (https://grouplens.org/datasets/movielens/25m/). Each cell of the data matrix represents the rating of a user (row) from 1 to 5 for the corresponding movie (column). In order to fit a Bernouli VAE we considered the matrix as binary i.e. if a user has seen a movie (1) or not (0). We then selected the 60 most popular movies, as well as, 10000 users who have seen between 2 and 30 of these movies. We also verified that all the movies have been seen from at least 600 users. In this way we reduced the size of the dataset, obtaining a realistic scenario where: 1) some movies are more popular than the others, and 2) we do not include users that have seen 0 or almost all the movies. We show in Fig. B.1 the number of views for each movie and the number of movies each user has seen. In Table B.6 we present the details for the Bernouli VAE.

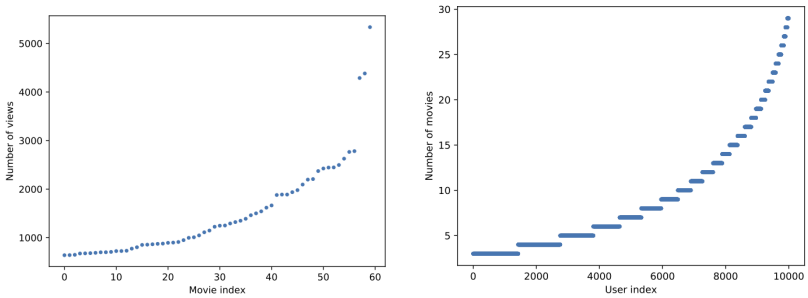


Figure B.1. The numbers of views for the movies and the users.

B.4.7 Information for fitting the LAND model

The locally adaptive normal distribution (LAND) [Arvanitidis et al., 2016b] is the extension of the normal distribution on Riemannian manifolds learned from data. Pennec [2006b] first derived this distribution on predefined manifolds as the sphere and also showed that it is the maximum entropy distribution given a mean and a precision matrix. The flexibility of this probability density relies on the shortest paths. However, the computational demand to fit this model is relatively high, especially in our case, since we need to use an approximation scheme to find the shortest paths.

In particular, we compute the *logarithmic map* $\boldsymbol{v} = \text{Log}_{\boldsymbol{x}}(\boldsymbol{y})$ by first finding the shortest path between $\boldsymbol{x}$ and $\boldsymbol{y}$, and then, rescaling the initial velocity as $\boldsymbol{v} = \frac{\dot{c}(0)}{||\dot{c}(0)||}\text{Length}(c)$, which ensures that $||\boldsymbol{v}|| = \text{Length}(c)$. In addition, for the estimation of the normalization constant we use the *exponential map* $\text{Exp}_{\boldsymbol{x}}(\boldsymbol{v}) = c_{\boldsymbol{v}}(t)$, which is the inverse operator that generates the shortest path with $c(1) = \boldsymbol{y}$ taking the rescaled initial velocity $\boldsymbol{v}$ as input. Also, we should be able to evaluate the metric. While

the logarithmic map can be approximated using our approach (Section B.4.2), for the exponential map we need to solve the ODEs system equation B.1 as an initial value problem (IVP). Note that we fit the LAND using gradient descent, which implies that the computation of these operators is the main computational bottleneck.

We provided a method in Proposition 4, which enables us to approximate the pullback metric in the latent space of a generative model using the corresponding KL divergence. Even if this is a sensible approach, in practice, the computational cost is relatively high as we might need to estimate the KL using Monte Carlo. For example, this is the case when the likelihood is the von Mises-Fisher. This further implies that fitting the LAND using this approach is prohibited due to the computational cost. Especially, since we need to evaluate many times the metric and its derivative for the computation of each exponential map. Hence, in order to fit the LAND efficiently, we used the following approximation based on Hauberg et al. [2012].

First we construct a uniformly spaced grid in the latent space. Then, we evaluate the metric using Proposition 4 for each point on the grid getting a set $\{\boldsymbol{z}_s, G_s\}_{s=1}^{S}$ of metric tensors. Thus, we can estimate the metric at any point $\boldsymbol{z}$ as

$$G(\boldsymbol{z}) = \sum_{s=1}^{S} \widetilde{w}_s(\boldsymbol{z}) G_s, \quad \text{with} \quad \widetilde{w}_s(\boldsymbol{z}) = \frac{w_s(\boldsymbol{z})}{\sum_{j=1}^{S} w_s(\boldsymbol{z})} \quad \text{and} \quad w_s(\boldsymbol{z}) = \exp\left(-\frac{||\boldsymbol{z}_s - \boldsymbol{z}||^2}{2\sigma^2}\right) \tag{B.22}$$

where $\sigma > 0$ the bandwidth parameter. This is by definition a Riemannian metric as a weighted sum of Riemannian metrics with a smooth weighting function. In this way, we can approximate the pullback of the Fisher-Rao metric in the latent space $\mathcal{Z}$ in order to perform the necessary computations more efficiently.

APPENDIX C

Appendix to chapter 4

C.1 Proof of the lower bound on the data manifold log likelihood

We will denote with $\mathcal{M}$ the data manifold of dimension d embedded in some higher dimensional Euclidean ambient space $\mathbb{R}^D$. An open cover $\mathcal{U}$ of $\mathcal{M}$ consists of K local coordinate charts $(U_i, \phi_i)_{i=1}^K$ with $U_i \subset \mathcal{M}$ and $\phi_i : U_i \to V_i \subset \mathbb{R}^d$. A probability density function $p_{\mathcal{M}} : \mathcal{M} \to \mathbb{R}$ can be constructed with a *smooth partition of unity subordinate to* $\mathcal{U}$. That is, an indexed family $\{f_i\}_{i=1}^K$ of smooth functions with $\text{supp} f_i \subset U_i$, where for a neighborhood around any data point $\boldsymbol{x} \in \mathcal{M}$, only a finite subset of $\{f_i\}$ is non-zero and $\sum_{i=1}^K f_i(x) = 1$. For our particular case, we take $f_i = w_i p_{U_i}$ with $w_i \in [0, 1]$ and construct $p_{\mathcal{M}}$ as a weighted sum of smooth density functions defined locally in each coordinate patch U_i, i.e. $p_{\mathcal{M}}(\boldsymbol{x}) = \sum_{i=1}^K w_i p_{U_i} = \sum_i^K w_i p_{V_i}(\boldsymbol{u}) \det |G_i(\boldsymbol{u})|^{-\frac{1}{2}}$, with p_{V_i} the base distribution in Euclidean subset V_i, $\boldsymbol{u} = \phi_i(\boldsymbol{x})$ and $G_i(\boldsymbol{u}) = J_{\phi_{i-1}}^T J_{\phi_i^{-1}}$.

Proposition 11. *The log-likelihood* $\log p_{\mathcal{M}}(\boldsymbol{x})$ *is bounded from below by* $\mathcal{L} = \log \left[C \cdot \sum_i^K w_i p_{V_i}(\boldsymbol{u}) Tr(J_{\phi_i^{-1}}^T(\boldsymbol{u}) J_{\phi_i^{-1}}(\boldsymbol{u}))^{-\frac{d}{2}} \right]$ *with* $C = d^{d/2}$

Proof For a local coordinate chart (U, ϕ), with $U \subset \mathcal{M}$ with $\mathcal{M} \subset \mathbb{R}^D$ and $\phi : \mathcal{M} \to \mathbb{R}^d$, we denote the log-likelihood in neighborhood U by $\log p_U(\boldsymbol{x})$. Furthermore, denoting the singular values of matrix $J_{\phi^{-1}}^T(\boldsymbol{u})$ by s_i, we will use Jensen's inequality to first lower-bound the probability density in a local neighborhood U:

$$\frac{1}{2}\sum_{i=1}^{d}\log s_i^2 = \frac{d}{2}\sum_{i=1}^{d}\frac{1}{d}\log s_i^2 \leq \frac{d}{2}\log\left(\sum_{i=1}^{d}\frac{s_i^2}{d}\right) = \frac{d}{2}\log\left(\sum_{i=1}^{d}s_i^2\right) - \frac{d\log(d)}{2} \tag{C.1}$$

$$-\frac{1}{2}\sum_{i=1}^{d}\log s_i^2 \geq -\frac{d}{2}\log\left(\sum_{i=1}^{d}s_i^2\right) + \frac{d\log(d)}{2} \tag{C.2}$$

$$\log p_V(\boldsymbol{u}) - \frac{1}{2}\sum_{i=1}^{d}\log s_i^2 \geq \log p_V(\boldsymbol{u}) - \frac{d}{2}\log\left(\sum_{i=1}^{d}s_i^2\right) + \frac{d\log(d)}{2} \tag{C.3}$$

$$\log p_V(\boldsymbol{u}) - \frac{1}{2}\log[J_{\phi^{-1}}^T(\boldsymbol{u})J_{\phi^{-1}}(\boldsymbol{u})] \geq \log p_V(\boldsymbol{u}) - \frac{d}{2}\log\left(\sum_{i=1}^{d}s_i^2\right) + \frac{d\log(d)}{2} \tag{C.4}$$

$$\log p_U(\boldsymbol{x}) \geq \log p_V(\boldsymbol{u}) - \frac{d}{2}\log\left(\sum_{i=1}^{d}s_i^2\right) + \frac{d\log(d)}{2} \tag{C.5}$$

Now for a matrix A, we have:

$$Tr(A^TA) = Tr(U^T\Sigma^TVV^T\Sigma U) = Tr(\Sigma^T\Sigma UU^T) = Tr(\Sigma^T\Sigma) = \sum_{i=1}^{d}s_i^2 \tag{C.6}$$

with U, V orthogonal matrices and Σ a diagonal matrix containing the singular values of A.

Thus, eq. C.5 becomes:

$$\log p_U(\boldsymbol{x}) \geq \log p_V(\boldsymbol{u}) - \frac{d}{2}\log Tr(J_{\phi^{-1}}^T(\boldsymbol{u})J_{\phi^{-1}}(\boldsymbol{u})) + \frac{d\log(d)}{2} = \mathcal{L}_U \tag{C.7}$$

Thus, we have introduced a lower bound to the probability density in neighborhood $U \in \mathcal{M}$. Because $\log(\cdot)$ is a monotonic function and for all i we have $w_i \in [0,1]$, the direction of the inequality in eq. C.5 is preserved for all K neighborhoods in our open cover of $\mathcal{M}$, so our lower bound holds for the complete data log likelihood:

$$\log p(\boldsymbol{x}) = \log\sum_{i}^{K}w_i p_{U_i}(\boldsymbol{x}) \tag{C.8}$$

$$= \log\sum_{i}^{K}w_i p_{V_i}(\boldsymbol{u})\det|G_i(\boldsymbol{u})|^{-\frac{1}{2}} \tag{C.9}$$

$$\geq \log\left[C\cdot\sum_{i}^{K}w_i p_{V_i}(\boldsymbol{u})Tr\left(J_{\phi_i^{-1}}^T(\boldsymbol{u})J_{\phi_i^{-1}}(\boldsymbol{u})\right)^{-\frac{d}{2}}\right] = \mathcal{L} \tag{C.10}$$

with $C = d^{d/2}$. Using Hutchinson's estimator we can compute the trace efficiently. With $J_{\phi^{-1}} \in \mathbb{R}^{D \times d}$ and $p(\boldsymbol{\epsilon}) = \mathcal{N}(0, I_D)$:

$$\mathcal{L}_U \approx \log p_{V_i}(\boldsymbol{u}) - \frac{d}{2} \log \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[\boldsymbol{\epsilon}^T J^T_{\phi^{-1}}(\boldsymbol{u}) J_{\phi^{-1}}(\boldsymbol{u}) \boldsymbol{\epsilon} \right] + \frac{d \log(d)}{2} \tag{C.11}$$

$$= \log p_{V_i}(\boldsymbol{u}) - \frac{d}{2} \log \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[||J^T_{\phi^{-1}} \boldsymbol{\epsilon}||_2^2 \right] + \frac{d \log(d)}{2} \tag{C.12}$$

C.2 Details on synthetic 2D experiments

Datasets For all target densities in Figure 4.3 we generated 50000 points for the train set and 10000 points for the validation set. For details on generating the datasets, see Lou et al. [2020].

Architectures Table C.1 shows the architecture details for MCF. All flow layers are implemented as rational quadratic coupling flows. In all cases, our base distributions p_{V_i} are standard normals over the Euclidean spaces V_i. The distribution of the orthogonal directions to the manifold $p_{V_i^\perp}$ is a standard normal with $\sigma^2 = 0.01$.

Baseline implementations are provided by Lou et al. [2020] in `https://github.com/CUAI/Neural-Manifold-Ordinary-Differential-Equations`.

Hyperparameters	Datasets	
	Checkerboard ($\mathbb{S}^2$)	Four wrapped Normals ($\mathbb{S}^2$)
Charts	4	2
Chart flow layers	3	2
Chart bins	5	1
Spline range	[-3, 3]	[-4, 4]
Linear transform	LU	LU
ResNet layers (& units)	2 (64)	2 (16)
Activation	ReLU	ReLU

Table C.1. Architecture details for MCF on all synthetic datasets.

Training We train MCF using maximum likelihood. For the *spherical checkerboard* dataset we train for 300 epochs and for the *four wrapped normals* we train for 250 epochs. For both datasets we used a batch size of 256 with a learning rate of $3 \cdot 10^{-4}$.

To train NCPS we used a learning rate of 10^{-3} and a batch size of 200. For the *spherical checkerboard* dataset we train for 10000 epochs, while for the *four wrapped normals* dataset we train for 5000 epochs.

For NMODE, on *four wrapped normals* we used a batch size of 200 and a learning rate of 10^{-2}, training for 600 epochs. For the *spherical checkerboard* we used a batch size of 200 and a learning rate of 10^{-2}, training for 700 epochs.

All models are trained with the Adam optimizer [Kingma and Ba, 2014]. In general, we chose baseline hyperparameters such that we can have the fastest possible convergence without sacrificing training stability. Please note however that this is a different training setting to the one used for NMODE and the other baselines by Lou et al. [2020], as they generated a random batch of points on the manifold for every iteration, whereas in our case we generate a fixed amount of training points and iterate on those. We think that while this is a much harder training scenario, it's also a more realistic one.

C.3 Details on real world 2D experiments

Hyperparameters	Datasets	
	Fires	*Earthquakes*
Charts	2	2
Chart flow layers	5	4
Chart bins	10	16
Spline range	[-6, 6]	[-6, 6]
Linear transform	LU	LU
ResNet layers (& units)	2 (100)	2 (64)
Activation	CELU	CELU

Table C.2. Architecture details for MCF on all synthetic datasets.

C.3.1 Experimental details

MCF For architectural details please see table C.2. Our base distribution p_{V_i} is a standard normal in the Euclidean spaces V_i. The distribution of the orthogonal directions to the manifold $p_{V_i^\perp}$ is a standard normal with $\sigma^2 = 0.01$.

Baselines The architectures of both the NMODE and NCPS baselines are the same as in the synthetic datasets case.

Datasets The *fires* dataset consists of 66444 data points, while the *earthquakes* dataset consists of 5883 data points at the time of writing. We shuffle both datasets and keep 80% for the training sets and 20% for the validation sets.

Training on *earthquakes* We train MCF using maximum likelihood for 3000 epochs using the Adam optimizer with a batch size of 128 and a learning rate of $3 \cdot 10^{-4}$. Throughout training we annealed the learning rate using a cosine annealing schedule.

We train NCPS with the Adam optimizer for 10000 epochs with a learning rate of 10^{-3} and a batch size of 200.

We train NMODE with the Adam optimizer for 10000 epochs with a learning rate $3 \cdot 10^{-4}$ and a batch size of 500.

Training on *fires* We train MCF using maximum likelihood for 1000 epochs using the Adam optimizer with a batch size of 256 and a learning rate 10^{-4}. Throughout training we annealed the learning rate using a cosine annealing schedule and clipped the gradient norm to 8.

We train NCPS with the Adam optimizer for 3000 epochs with a learning rate of 10^{-3} and a batch size of 200.

We train NMODE with the Adam optimizer for 600 epochs with a learning rate $3{\cdot}10^{-4}$ and a batch size of 500.

For all baselines in both datasets we decay the learning rate every 1/3d of the total epochs with a scaling factor of 0.1. We found that training was difficult for all models. The hardest model to train was NMODE even though results for both baselines are generally unsatisfactory (see figure C.1). Given this, in our choice of hyperparameters we attempted to strike a balance between fast convergence and stable gradient updates. Furthermore, we checkpoint all models to retain the best performing parameter configuration according to validation results.

C.4 Details on the Lorenz experiment

C.4.1 Architecture

MCF Our model uses two coordinate charts to parameterize the manifold. The chart models ϕ comprise five flow layers. These are implemented as rational quadratic coupling layers, interspersed with random feature permutations. We use five bins in the range $[-3, 3]$. Each coupling transform is parameterized by a residual network with 1 residual block containing 2 hidden layers per block. Each hidden layer consists of 32 ReLU units. Our base distribution p_{V_i} is a standard normal over the Euclidean spaces V_i. The distribution of the orthogonal directions to the manifold $p_{V_i^\perp}$ is a standard normal with $\sigma^2 = 0.01$.

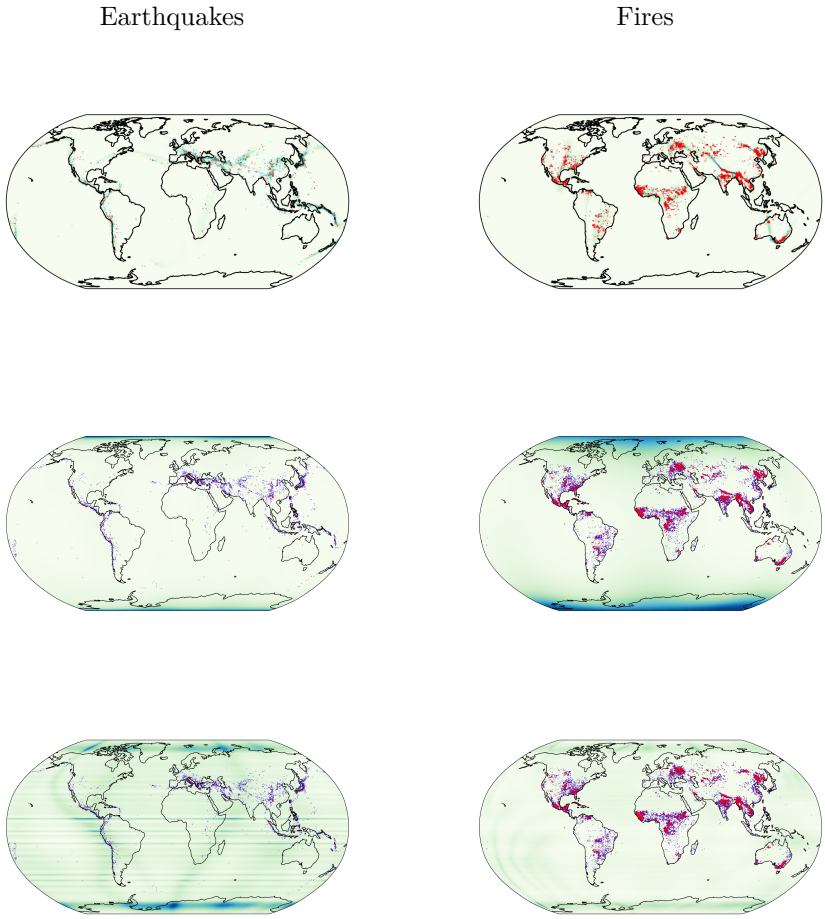


Figure C.1. Density estimation results on the earthquakes and fires data. Robinson projection. Top row: *MCF* (ours), middle row: *NMODE*, bottom row: *NCPS*

$\mathcal{M}$-flow For $\mathcal{M}$-flow we reproduced the reference architecture given by Brehmer and Cranmer [2020]. Both the chart model and the base model comprise 5 rational quadratic coupling layers, interspersed with random feature permutations. We use 5 bins for both maps in the range [-3, 3]. Each coupling transform is parameterized by a residual network with 2 residual blocks and 2 hidden layers per block. Each hidden layer consists of 100 ReLU units.

C.4.2 Training

MCF We trained the model on a dataset of 10^6 samples using maximum likelihood training for 1000 epochs. The AdamW optimizer [Loshchilov and Hutter, 2017] was used with a learning rate of 10^{-4}. We use a batch size of 10000.

$\mathcal{M}$-flow We trained the model on a dataset of 10^6 samples with split manifold learning and maximum likelihood training phases, assigning 50 epochs to each phase (100 in total). The AdamW optimizer was used with a learning rate of $3 \cdot 10^{-4}$, cosine annealing and weight decay of 10^{-4}. We use a batch size of 100.

C.5 Details on the Large Hadron Collider experiment

For details on dataset generation, as well as an explanation on the closure tests we refer the interested reader to Brehmer and Cranmer [2020]. The dataset itself can be found in https://drive.google.com/drive/folders/13x8lEO8--L8-ORoN_QTUbSC_fRBAdRPT.

MCF Our model uses five coordinate charts to parameterize the manifold. The chart models ϕ comprise ten flow layers. These are implemented as rational quadratic coupling layers, interspersed with LU-decomposed invertible linear transformations. We use 11 bins in the range [-10, 10]. Each coupling transform is parameterized by a residual network with two residual blocks of two hidden layers per block. Each hidden layer consists of 100 ReLU units. Our base distributions p_{V_i} are standard normals over the Euclidean spaces V_i. The distribution of the orthogonal directions to the manifold $p_{V_i^\perp}$ is a standard normal with $\sigma^2 = 0.01$.

Baselines To estimate baseline runtimes we run both RQ-flow and $\mathcal{M}$-flow but we note that baseline results are taken from the paper itself. Both baselines are composed of 35 rational quadratic coupling layers, interspersed with LU-decomposed invertible linear transformations. For $\mathcal{M}$-flow, the chart model ϕ uses 20 layers and the base model h uses 15 layers. Each coupling transform is parameterized by a residual network with two residual blocks of two hidden layers per block. Each hidden layer consists of 100 ReLU units. All runtime estimations are based on the implementation provided by Brehmer and Cranmer [2020], which can be found in https://github.com/johannbrehmer/manifold-flow.

Training We trained our model using maximum likelihood on the same dataset as Brehmer and Cranmer [2020], using 10^6 samples. We used the AdamW optimizer

with a learning rate of $3 \cdot 10^{-4}$, a batch size of 256, cosine annealing and a weight decay of 10^{-5} and trained the model for 50 epochs.

Evaluation Our evaluation procedure is identical to Brehmer and Cranmer [2020]. In brief, we generate 3 different datasets using 3 different parameter points $\theta_1 = (0, 0), \theta_2 = (0.5, 0)$ and $\theta_3 = (-1, -1)$. Each dataset has 15 i.i.d. samples. For each model and each observed dataset, we generate four MCMC chains of length 750 each, with a Gaussian proposal distribution with mean step size 0.15 and a burn in of 100 steps. Then we obtain kernel density estimates of the log-posterior for each of the 3 parameter points and report the average value in table 4.1. Like Brehmer and Cranmer [2020] we train 5 instances of our model with independent initializations, remove the top and bottom value and report the mean over the remaining runs.

Bibliography

Shunichi Amari. *Information Geometry and Its Applications*. Springer Publishing Company, Incorporated, 1st edition, 2016. ISBN 4431559779.

Georgios Arvanitidis, Lars K Hansen, and Søren Hauberg. A locally adaptive normal distribution. In *Advances in Neural Information Processing Systems*, pages 4251–4259, 2016a.

Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. A locally adaptive normal distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, jun 2016b.

Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. In *International Conference on Learning Representations (ICLR)*, 2018.

Georgios Arvanitidis, Søren Hauberg, Philipp Hennig, and Michael Schober. Fast and robust shortest paths on manifolds learned from data. In *Proceedings of the 19th international Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Georgios Arvanitidis, Miguel González-Duque, Alison Pouplin, Dimitris Kalatzis, and Søren Hauberg. Pulling back information geometry. *arXiv preprint arXiv:2106.05367*, 2021a.

Georgios Arvanitidis, Søren Hauberg, and Bernhard Schölkopf. Geometrically Enriched Latent Spaces. In *Artificial Intelligence and Statistics (AISTATS)*, 2021b.

Matthias Bauer and Andriy Mnih. Resampled priors for variational autoencoders. *arXiv preprint arXiv:1810.11428*, 2018.

Hadi Beik-Mohammadi, Søren Hauberg, Georgios Arvanitidis, Gerhard Neumann, and Leonel Rozo. Learning riemannian manifolds for geodesic motion skills. In *Robotics: Science and Systems (R:SS)*, 2021.

Jan Jetze Beitler, Ivan Sosnovik, and Arnold Smeulders. Pie: Pseudo-invertible encoder. *arXiv preprint arXiv:2111.00619*, 2021.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Wouter Boomsma, Kanti V Mardia, Charles C Taylor, Jesper Ferkinghoff-Borg, Anders Krogh, and Thomas Hamelryck. A generative, probabilistic model of local protein structure. *Proceedings of the National Academy of Sciences*, 105(26):8932–8937, 2008.

Joey Bose, Ariella Smofsky, Renjie Liao, Prakash Panangaden, and Will Hamilton. Latent variable modelling with hyperbolic normalizing flows. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1045–1055, Virtual, 13–18 Jul 2020. PMLR. URL http://proceedings.mlr.press/v119/bose20a.html.

Matthew Brand. Charting a manifold. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 961–968. MIT Press, 2002. URL https://proceedings.neurips.cc/paper/2002/hash/8929c70f8d710e412d38da624b21c3c8-Abstract.html.

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *arXiv preprint arXiv:2003.13913*, 2020.

Johann Brehmer, Gilles Louppe, Juan Pavez, and Kyle Cranmer. Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences*, 117(10):5242–5249, 2020.

Alice Le Brigant and Stéphane Puechmorel. The fisher-rao geometry of beta distributions applied to the study of canonical moments, 2019.

Richard L Burden, J Douglas Faires, and Annette M Burden. *Numerical analysis*. Cengage learning, 2015.

Nutan Chen, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick Smagt. Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1550, 2018a.

Nutan Chen, Alexej Klushyn, Alexandros Paraschos, Djalel Benbouzid, and Patrick van der Smagt. Active learning based on data uncertainty and model sensitivity, 2018b.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018c.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

Earl A Coddington and Norman Levinson. *Theory of ordinary differential equations*. Tata McGraw-Hill Education, 1955.

Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2133–2143, Virtual, 13–18 Jul 2020. PMLR. URL http://proceedings.mlr.press/v119/cornish20a.html.

Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, 2018a.

Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. Hyperspherical variational auto-encoders. *34th Conference on Uncertainty in Artificial Intelligence (UAI-18)*, 2018b.

Nicki S Detlefsen, Martin Jørgensen, and Søren Hauberg. Reliable training and estimation of variance networks. *arXiv preprint arXiv:1906.03260*, 2019a.

Nicki S Detlefsen, Martin Jørgensen, and Søren Hauberg. Reliable training and estimation of variance networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019b.

Nicki Skafte Detlefsen, Søren Hauberg, and Wouter Boomsma. What is a meaningful representation of protein sequences?, 2020.

Laurent Dinh, Jascha Sohl-Dickstein, Razvan Pascanu, and Hugo Larochelle. A RAD approach to deep mixture models. *CoRR*, abs/1903.07714, 2019. URL http://arxiv.org/abs/1903.07714.

M.P. do Carmo. *Riemannian Geometry*. Mathematics (Boston, Mass.). Birkhäuser, 1992.

Hadi Mohaghegh Dolatabadi, Sarah Erfani, and Christopher Leckie. Invertible generative modeling using linear rational splines. In *International Conference on Artificial Intelligence and Statistics*, pages 4236–4246. PMLR, 2020.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 3140–3150. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/21be9a4bd4f81549a9d1d241981cec3c-Paper.pdf.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 7511–7522. Curran Associates, Inc., 2019a. URL https://proceedings.neurips.cc/paper/2019/file/7ac71d433f282034e088473244df8c02-Paper.pdf.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Cubic-spline flows. *arXiv preprint arXiv:1906.02145*, 2019b.

David Eklund and Søren Hauberg. Expected path length on random manifolds. *arXiv preprint arXiv:1908.07377*, 2019.

EOSDIS. Active fire data. https://earthdata.nasa.gov/earth-observation-data/near-real-time/firms/active-fire-data, 2020. Land, Atmosphere Near real-time Capability for EOS (LANCE) system operated by NASA's Earth Science Data and Information System (ESDIS).

Luca Falorsi and Patrick Forré. Neural ordinary differential equations on manifolds. *arXiv preprint arXiv:2006.06663*, 2020.

Luca Falorsi, Pim de Haan, Tim R Davidson, Nicola De Cao, Maurice Weiler, Patrick Forré, and Taco S Cohen. Explorations in homeomorphic variational auto-encoding. *arXiv preprint arXiv:1807.04689*, 2018.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

Thomas Fletcher. Geodesic regression on riemannian manifolds. In *Proceedings of the Third International Workshop on Mathematical Foundations of Computational Anatomy-Geometrical and Statistical Methods for Modelling Biological Shape Variability*, pages 75–86, 2011.

Sylvestre Gallot, Dominique Hulin, and Jacques Lafontaine. *Riemannian metrics*, pages 51–127. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

Mevlana C Gemici, Danilo Rezende, and Shakir Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.

Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Thomas Hamelryck, John T Kent, and Anders Krogh. Sampling realistic protein conformations using local structural bias. *PLoS Comput Biol*, 2(9):e131, 2006.

Søren Hauberg. Principal Curves on Riemannian Manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016.

Søren Hauberg. Directional statistics with the spherical normal distribution. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 704–711. IEEE, 2018a.

Søren Hauberg. Only bayes should learn a manifold (on the estimation of differential geometric structure from data). *arXiv preprint arXiv:1806.04994*, 2018b.

Søren Hauberg. Only bayes should learn a manifold. 2018c.

Søren Hauberg, Oren Freifeld, and Michael J. Black. A Geometric Take on Metric Learning. In *Advances in Neural Information Processing Systems (NIPS) 25*, pages 2033–2041, 2012.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.123. URL https://doi.org/10.1109/ICCV.2015.123.

Philipp Hennig and Søren Hauberg. Probabilistic solutions to differential equations and their application to riemannian statistics. In *Proceedings of the 17th international Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 33, 2014.

Thomas Hillen, Kevin Painter, Amanda Swan, and Albert Murtha. Moments of von mises and fisher distributions and applications. *Mathematical Biosciences and Engineering*, 14:673–694, 12 2016. doi: 10.3934/mbe.2017038.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019.

Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, page 2, 2016.

Elton P Hsu. *Stochastic analysis on manifolds*, volume 38. American Mathematical Soc., 2002.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Dimitris Kalatzis, David Eklund, Georgios Arvanitidis, and Søren Hauberg. Variational autoencoders with riemannian brownian motion priors. *arXiv preprint arXiv:2002.05227*, 2020.

Dimitris Kalatzis, Johan Ziruo Ye, Alison Pouplin, Jesper Wohlert, and Søren Hauberg. Density estimation on smooth manifolds with normalizing flows. *arXiv preprint arXiv:2106.03500*, 2021.

Anuj Karpatne, Imme Ebert-Uphoff, Sai Ravela, Hassan Ali Babaie, and Vipin Kumar. Machine learning for the geosciences: Challenges and opportunities. *IEEE Transactions on Knowledge and Data Engineering*, 31(8):1544–1554, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

Alexej Klushyn, Nutan Chen, Richard Kurle, Botond Cseke, and Patrick van der Smagt. Learning hierarchical priors in vaes. In *Advances in Neural Information Processing Systems*, pages 2866–2875, 2019.

Samuli Laine. Feature-based metrics for exploring the latent space of generative models. 2018.

Neil Lawrence. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *J. Mach. Learn. Res.*, 2005a.

Neil D. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of machine learning research*, 6(Nov):1783–1816, 2005b.

John M Lee. Smooth manifolds. In *Introduction to Smooth Manifolds*, pages 1–31. Springer, 2013.

Henry Li, Ofir Lindenbaum, Xiuyuan Cheng, and Alexander Cloninger. Variational diffusion autoencoders with random walk sampling. *arXiv preprint arXiv:1905.12724*, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Aaron Lou, Derek Lim, Isay Katsman, Leo Huang, Qingxuan Jiang, Ser-Nam Lim, and Christopher De Sa. Neural manifold ordinary differential equations. *arXiv preprint arXiv:2006.10254*, 2020.

Kanti V Mardia and Peter E Jupp. Basic concepts and models. *Mardia KV, Jupp PE. Directional statistics, 2nd edition. Chichester (UK): John Wiley & Sons*, pages 25–56, 2000.

James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

Emile Mathieu and Maximilian Nickel. Riemannian continuous normalizing flows. *arXiv preprint arXiv:2006.10605*, 2020.

Emile Mathieu, Charline Le Lan, Chris J Maddison, Ryota Tomioka, and Yee Whye Teh. Continuous hierarchical representations with poincaré variational auto-encoders. In *Advances in neural information processing systems*, pages 12544–12555, 2019.

Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Trans. Graph.*, 38(5), October 2019. ISSN 0730-0301. doi: 10.1145/3341156. URL https://doi.org/10.1145/3341156.

Yoshihiro Nagano, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama. A differentiable gaussian-like distribution on hyperbolic space for gradient-based learning. *arXiv preprint arXiv:1902.02992*, 2019.

Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders. *arXiv preprint arXiv:1605.06197*, 2016.

Jacinto C. Nascimento, Jorge G. Silva, Jorge S. Marques, and João Miranda Lemos. Manifold learning for object tracking with multiple nonlinear models. *IEEE Trans. Image Process.*, 23(4):1593–1605, 2014. doi: 10.1109/TIP.2014.2303652. URL https://doi.org/10.1109/TIP.2014.2303652.

Frank Nielsen. An elementary introduction to information geometry. *Entropy*, 22(10): 1100, Sep 2020. ISSN 1099-4300. doi: 10.3390/e22101100. URL http://dx.doi.org/10.3390/e22101100.

NOAA. Ncei/wd5 global significant earthquake database. https://www.ngdc.noaa.gov/hazard/earthqk.shtml, 2020. National Geophysical Data Center / World Data Service (NGDC/WDS).

Josep M Oller. On an intrinsic analysis of statistical estimation. In *Multivariate Analysis: Future Directions 2*, pages 421–437. Elsevier, 1993.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017.

George Papamakarios, Eric T. Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22:57:1–57:64, 2021. URL http://jmlr.org/papers/v22/19-1028.html.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*. 2019.

David Peel, William J Whiten, and Geoffrey J McLachlan. Fitting mixtures of kent distributions to aid in joint set identification. *Journal of the American Statistical Association*, 96(453):56–63, 2001.

Xavier Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127, 2006a.

Xavier Pennec. Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements. *Journal of Mathematical Imaging and Vision*, 2006b.

Nikolaos Pitelis, Chris Russell, and Lourdes Agapito. Learning a manifold as an atlas. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 1642–1649. IEEE Computer Society, 2013. doi: 10.1109/CVPR.2013.215. URL https://doi.org/10.1109/CVPR.2013.215.

Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC press, 1987.

Qichao Que and Mikhail Belkin. Back to the future: Radial basis function networks revisited. In *Artificial Intelligence and Statistics (AISTATS)*, 2016.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014.

Luis A Pérez Rey, Vlado Menkovski, and Jacobus W Portegies. Diffusion variational autoencoders. *arXiv preprint arXiv:1901.08991*, 2019.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/rezende15.html.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and variational inference in deep latent gaussian models. In *International Conference on Machine Learning*, volume 2, 2014.

Danilo Jimenez Rezende, George Papamakarios, Sebastien Racaniere, Michael Albergo, Gurtej Kanwar, Phiala Shanahan, and Kyle Cranmer. Normalizing flows on tori and spheres. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8083–8092, Virtual, 13–18 Jul 2020. PMLR. URL http://proceedings.mlr.press/v119/rezende20a.html.

Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 1999.

Mihaela Rosca, Balaji Lakshminarayanan, and Shakir Mohamed. Distribution matching in variational inference. *arXiv preprint arXiv:1802.06847*, 2018.

Daniel M Roy, Charles Kemp, Vikash K Mansinghka, and Joshua B Tenenbaum. Learning annotated hierarchies from relational data. In *Advances in neural information processing systems*, pages 1185–1192, 2007.

Noam Rozen, Aditya Grover, Maximilian Nickel, and Yaron Lipman. Moser flow: Divergence-based generative modeling on manifolds. *Advances in Neural Information Processing Systems*, 34, 2021.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986a.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323, 1986b.

Aidan Scannell, Carl Henrik Ek, and Arthur Richards. Trajectory Optimisation in Learned Multimodal Dynamical Systems Via Latent-ODE Collocation. In *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2021.

Stefan Schonsheck, Jie Chen, and Rongjie Lai. Chart auto-encoders for manifold structured data. *arXiv preprint arXiv:1912.10094*, 2019.

Hang Shao, Abhishek Kumar, and P Thomas Fletcher. The riemannian geometry of deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 315–323, 2018.

Ondrej Skopek, Octavian-Eugen Ganea, and Gary Bécigneul. Mixed-curvature variational autoencoders. *arXiv preprint arXiv:1911.08411*, 2019.

Mark Steyvers and Joshua B Tenenbaum. The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive science*, 29 (1):41–78, 2005.

Andrew Stirn and David A Knowles. Variational variance: Simple and reliable predictive variance parameterization. *arXiv preprint arXiv:2006.04910*, 2020.

Robert S Strichartz. *A guide to distribution theory and Fourier transforms*. World Scientific Publishing Company, 2003.

Jakub Tomczak. Fisher information matrix for Gaussian and categorical distributions. https://www.ii.pwr.edu.pl/~tomczak/PDF/[JMT]Fisher_inf.pdf, 2012. Online; accessed 17 Mai 2021.

Jakub M Tomczak and Max Welling. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*, 2017.

Alessandra Tosi, Søren Hauberg, Alfredo Vellido, and Neil D. Lawrence. Metrics for Probabilistic Geometries. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, July 2014.

Maxime Tournier, Xiaomao Wu, Nicolas Courty, Elise Arnaud, and Lionel Reveret. Motion compression using principal geodesics analysis. In *Computer Graphics Forum*, volume 28, pages 355–364. Wiley Online Library, 2009.

Warwick Tucker. A rigorous ode solver and smale's 14th problem. *Foundations of Computational Mathematics*, 2(1):53–117, 2002.

Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. *Advances in neural information processing systems*, 32, 2019.

Jiacheng Xu and Greg Durrett. Spherical latent spaces for stable variational autoencoders. *arXiv preprint arXiv:1808.10805*, 2018.

Tao Yang, Georgios Arvanitidis, Dongmei Fu, Xiaogang Li, and Søren Hauberg. Geodesic clustering in deep generative models. In *arXiv preprint*, 2018.

Miaomiao Zhang and Tom Fletcher. Probabilistic principal geodesic analysis. *Advances in Neural Information Processing Systems*, 26:1178–1186, 2013.

{2.7182818284}ϑφερτυθιοπσδφγηξκλ